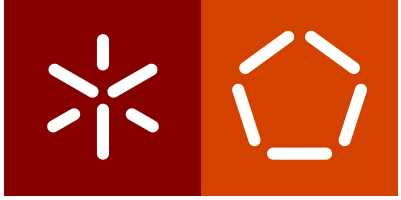


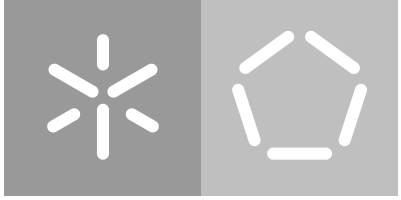
University of Minho

School of Engineering

Carla Rafaela Mendes da Silva

Development of a framework for the classification of antibiotics adjuvants

October 2022



University of Minho

School of Engineering

Carla Rafaela Mendes da Silva

Development of a framework for the classification of antibiotics adjuvants

Master's dissertation

Master's in BioInformatics

Dissertation supervised by

Oscar Manuel Lima Dias

Ana Margarida Rodrigues Sousa

October 2022

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos. Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada. Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Atribuição-CompartilhaIgual

CC BY-SA

https://creativecommons.org/licenses/by-sa/4.0/

Dedicado aos meus pais Carla e Eduardo
e ao meu irmão, Miguel

ACKNOWLEDGEMENTS

Throughout my academic years, some people helped me surpass my difficulties, so I would like to express my gratitude for their support and cooperation throughout the realisation of the bachelor's and master's degrees, especially this dissertation.

Firstly a very particular thanks to my supervisors, Oscar Manuel Lima Dias and Ana Margarida Rodrigues Sousa, for their work and help, which were essential to completing this dissertation's objectives. I would like to thank my friends for all the support and motivation transmitted in unhappy and happy times through the years. Also, I would like to thank my work teammates, who always motivated me to pursue my academic objectives.

Last but not least, I am incredibly grateful to my family, especially my parents and brother, who always supported, motivated, and encouraged me not to give up on my dreams and objectives.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

ABSTRACT

Throughout the last decades, bacteria have become increasingly resistant to available antibiotics, leading to a growing need for new antibiotics and new drug development methodologies. In the last 40 years, there are no records of the development of new antibiotics, which has begun to shorten possible alternatives. Therefore, finding new antibiotics and bringing them to market is increasingly challenging. One approach is finding compounds that restore or leverage the activity of existing antibiotics against biofilm bacteria. As the information in this field is very limited and there is no database regarding this theme, machine learning models were used to predict the relevance of the documents regarding adjuvants.

In this project, the ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application was developed to help researchers save time in their daily research. This application was constructed using Django and Django REST Framework for the backend and React for the frontend.

As for the backend, a database needed to be constructed since no database entirely focuses on this topic. For that, a machine learning model was trained to help us classify articles. Three different algorithms were used, Support-Vector Machine (SVM), Random Forest (RF), and Logistic Regression (LR), combined with a different number of features used, more precisely, 945 and 1890. When analyzing all metrics, model LR-1 performed the best for classifying relevant documents with an accuracy score of 0.8461, a recall score of 0.6170, an f1-score of 0.6904, and a precision score of 0.7837. This model is the best at correctly predicting the relevant documents, as proven by the higher recall score compared to the other models. With this model, our database was populated with relevant information.

Our backend has a unique feature, the aggregation feature constructed with Named Entity Recognition (NER). The goal is to identify specific entity types, in our case, it identifies

CHEMICAL and DISEASE. An association between these entities was made, thus delivering the user the respective associations, saving researchers time. For example, a researcher can see with which compounds "*pseudomonas aeruginosa*" has already been tested thanks to this aggregation feature.

The frontend was implemented so the user could access this aggregation feature, see the articles present in the database, use the machine learning models to classify new documents, and insert them in the database if they are relevant.

Keywords: Machine Learning, Feature Selection, Feature Generation, Adjuvants, Biofilms, Antimicrobial, Django, Django REST Framework, React

RESUMO

Ao longo das últimas décadas, as bactérias tornaram-se cada vez mais resistentes aos antibióticos disponíveis, levando a uma crescente necessidade de novos antibióticos e novas metodologias de desenvolvimento de medicamentos. Nos últimos 40 anos, não há registos do desenvolvimento de novos antibióticos, o que começa a reduzir as alternativas possíveis. Portanto, criar novos antibióticos e torna-los disponíveis no mercado é cada vez mais desafiante. Uma abordagem é a descoberta de compostos que restaurem ou potencializem a atividade dos antibióticos existentes contra bactérias multirresistentes. Como as informações neste campo são muito limitadas e não há uma base de dados sobre este tema, modelos de *Machine Learning* foram utilizados para prever a relevância dos documentos acerca dos adjuvantes.

Neste projeto, foi desenvolvida a aplicação ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* para ajudar os investigadores a economizar tempo nas suas pesquisas. Esta aplicação foi construída usando o Django e Django REST Framework para o *backend* e React para o *frontend*.

Quanto ao *backend*, foi necessário construir uma base de dados, pois não existe nenhuma que se concentre inteiramente neste tópico. Para isso, foi treinado um modelo *machine learning* para nos ajudar a classificar os artigos. Três algoritmos diferentes foram usados: Support-Vector Machine (SVM), Random Forest (RF) e Logistic Regression (LR), combinados com um número diferente de *features*, mais precisamente, 945 e 1890. Ao analisar todas as métricas, o modelo LR-1 teve o melhor desempenho para classificar artigos relevantes com uma *accuracy* de 0,8461, um *recall* de 0,6170, um *f1-score* de 0,6904 e uma *precision* de 0,7837. Este modelo foi o melhor a prever corretamente os artigos relevantes, comprovado pelo alto *recall* em comparação com os outros modelos. Com este modelo, a base de dados foi populda com informação relevante.

O *backend* apresenta uma caracteristica particular, a agregação construída com *Named-Entity-Recognition* (NER). O objetivo é identificar tipos específicos de entidades, no nosso caso, identifica QUÍMICOS e DOENÇAS. Esta classificação serviu para formar associações entre entidades, demonstrando ao utilizador as respetivas associações feitas, permitindo economizar o tempo dos investigadores. Por exemplo, um investigador pode ver com quais compostos a "*pseudomonas aeruginosa*" já foi testada graças à funcionalidade de agregação.

O *frontend* foi implementado para que o utilizador possa ter acesso a esta funcionalidade de agregação, ver os artigos presentes na base de dados, utilizar o modelo de *machine learning* para classificar novos artigos e inseri-los na base de dados caso sejam relevantes.

Palavras-chave: *Machine Learning*, *Feature Selection*, *Feature Generation*, Adjuvantes, Biofilmes, Antimicrobiaanos, Django, Django REST Framework, React

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

A

ACID Atomicity, Consistency, Isolation e Durability.

AI Artificial Intelligence.

AMR Antimicrobial Resistance.

ANOVA Analysis of Variance.

API Application Programming Interfaces.

AUC Area Under the Curve.

B

BIODR Biomedical Document Retrieval.

BIOTM Biomedical Text Mining.

C

CAZY Carbohydrate-Active Enzymes.

CRUD Create, Read, Update and Delete.

CSV Comma-Separated Values.

CV Cross Validation.

D

DB Database.

DBMS Database Management System.

DDBJ DNA Database of Japan.

DDL Data Definition Language.

DML Data Manipulation Language.

DNA Deoxyribonucleic Acid.

DOM Document Object Model.

DRF Django REST Framework.

E

EMBL European Molecular Biology Laboratory.

EMBL-EBI European Molecular Biology Laboratory - European Bioinformatics Institute.

F

FN False Negative.

FP False Positives.

FPR False Positive Rate.

G

GEO Gene Expression Omnibus.

H

HATEOAS Hypermedia as the engine of application state.

HTML HyperText Markup Language.

HTTP Hypertext Transfer Protocol.

I

IE Information Extraction.

IR Information Retrieval.

J

JS JavaScript.

JSON JavaScript Object Notation.

JSON-RCP JavaScript Object Notation-Remote Procedure Calls.

JSX JavaScript XML.

K

KDT Knowledge-Discovery in Text.

KEGG Kyoto Encyclopedia of Genes and Genomes.

L

LR Logistic Regression.

M

MEDLINE Medical Literature Analysis and Retrieval System Online.

MESH Medical Subject Headings.

MIC Minimum Inhibitory Concentrations.

ML Machine Learning.

MRNA Messenger RNA.

MRSA Methicillin Resistant *Staphylococcus aureus*.

N

NASA National Aeronautics and Space Administration.

NCBI National Centre for Biotechnology Information.

NCI-PID National Cancer Institute-Pathway Interaction Database.

NE Named Entities.

NER Named-entity Recognition.

NLP Natural Language Processing.

NLTK Natural Language Processing Toolkit.

O

OAUTH Open Authorization.

P

PBPS Penicillin-Binding Proteins.

PCA Principal Component Analysis.

PDBE Protein Data Bank in Europe.

PDBJ Protein Data Bank Japan.

PMID PubMed Unique Identifier Number.

PR Precision-Recall.

R

REST REpresentational State Transfer.

RF Random Forest.

RNA Ribonucleic Acid.

ROC Receiver Operating Characteristic.

S

SMD Stanford Microarray Database.

SMTP Simple Mail Transfer Protocol.

SOAP Simple Object Access Protocol.

SPA Single Page Application.

SQL Structed Query Language.

SVM Support Vector Machines.

T

T-SNE t-Distributed Stochastic Neighbor Embedding.

TF-IDF Term Frequency - Inverse Document Frequency.

TM Text Mining.

TN True Negatives.

TP True Positives.

TPR True Positive Rate.

U

UE User Experience.

UI User Interface.

URI Uniform Resource Identifier.

URL Uniform Resource Locator.

URLCONF URL Configuration.

UX User Experience.

V

VDOM Virtual Document Object Model.

W

WHO Worlds Health Organization.

WWPDB Worldwide Protein Data Bank.

WWW World Wide Web.

X

XML Extensible Markup Language.

XML-RPC Extensible Markup Language-Remote Procedure Calls.

XSS Cross-Site Scripting.

1

INTRODUCTION

1.1 CONTEXT

Alexander Fleming isolated penicillin in 1928, and since then, antibiotics have been one of the most important discoveries to health. However, the growing *AMR* led to times in which antibiotics are not permanently effective. The microbial resistance to antibiotics affects humans, animals, and the environment, leading to a global health care crisis that is the main challenge to health care in the 21st century.

Bacteria resist against antibiotics using distinct mechanisms, including the formation of biofilms. Biofilms are well-organised structures of microorganisms attached to biotic or abiotic surfaces and whose cells are encased and protected by a self-produced polymeric matrix. Bacteria within biofilms can exhibit 10 to 1000-fold higher tolerance to antimicrobial agents than their planktonic (or free-living) counterparts. Due to antimicrobials and immune defences resistance, up to 80% of the microbial infections associated with biofilms are considered chronic.

Antibiotics rarely have effect against biofilms which led the clinical community to seek alternative approaches of improving the in-use antibiotic's action. A way to mitigate the gap between new drugs and the supply pipeline is finding antibiotic adjuvants or' helper compounds' that boost their activity against biofilms. Compounds that have the ability to prevent biofilm formation, disrupt, or disperse biofilm cells to planktonic state might have the potential to function as antibiotic adjuvants. Over the past years much effort has been

put in to finding antibiofilm compounds, but no antibiofilm compounds reached to market. All the research of seeking antibiofilm compounds is dispersed in the literature, and there is no web-based central resource that allows storage, annotation, comparison and analysis of results, or even tracking the drug development stages making the search exhaustive and time-consuming. In this project, it will apply machine learning algorithms to systematise the collection of this information into a public access bioinformatics database, offering an inexistent resource to researchers search for the literature and obtain information about the antibiotic adjuvants.

1.2 MOTIVATION

The inefficacy of antibiotics to eradicate biofilms forces the community to seek alternative therapeutic strategies to improve the action of the in-use antibiotics. The combination between antibiofilm compounds and antibiotics has proven to be a winning strategy in fighting antibiotic-resistant bacteria and biofilms. However, there are no drug candidates for approval, which is rather disappointing given the increased number of publications about antibiofilm compounds. In light of these findings, the main motivation of this work is to significantly change this scenario by providing researchers with a bioinformatics tool that addresses the issues of developing an antibiofilm therapy.

1.3 GOALS

This project aims to develop a framework for the classification of antibiotics adjuvants. More precisely, this can be divided into several objectives:

- Literature review on ML applied to text mining, NER, biological databases, and other bioinformatics frameworks.
- Develop ML models to identify relevant bibliographic references for discovering antibiotics adjuvants.

- Create the dataset of relevant references and apply annotation using NER to identify possible antibiotic adjuvants.
- Develop a biological database to collect structured information from the previous steps.
- Develop a graphical user interface to provide non-bioinformaticians access to the database.

The long-term objective is to develop a framework to find information on antibiotic adjuvants. Preliminary studies showed that the information available on this topic is dispersed. Hence, the development of this framework is of the utmost importance, as it aims to collect all the dispersed information about antibiotics adjuvants available on a website ubiquitously accessible.

1.4 DISSERTATION OVERVIEW

The development of this project is divided into 5 phases, beginning with the state of the art, followed by the workflow implementation, their respective results, and a conclusion about all the work done.

Chapter 2 will focus on general subjects such as Bioinformatics, Databases, Text Mining, ML, implementation of other frameworks, Antibiotics resistance, and Biofilms.

- For Bioinformatics, relevant concepts in the field will be covered systematically.
- For ML, the difference between supervised, unsupervised and reinforcement Learning, and the assessment of ML models will be explained.
- A brief overview of biological databases, Text Mining, and implementation of other frameworks will be provided.
- For Antibiotic Resistance and Biofilms, the themes will be introduced and the central points will be explained

Chapter 3 details our methods for implementing the machine learning model, server, and client side. Initially, the construction of the model is described with the specified strategy. Next, a description of the server and client side is provided.

Chapter 4 presents the results of the implementation of this project and the respective discussion.

In **chapter 5**, we highlight our conclusions about the developed work, emphasize some of our limitations in the development of this project and discuss future work.

2

STATE OF THE ART

2.1 BIOINFORMATICS

2.1.1 *What is Bioinformatics?*

Contrary to popular belief, bioinformatics studies started more than 50 years ago when desktop computers were a hypothesis, and *DNA* was yet to be sequenced [1; 2]. Let's begin with the early 1950s, where there was little knowledge about *DNA*, but scientists speculated that a molecule could carry genetic information. In 1944, Avery, MacLeod, and McCarty proposed that the uptake of pure *DNA* from a virulent bacterial strain could confer virulence to a nonvirulent strain. However, the scientific community did not immediately recognise this as most people thought that proteins were the carriers of genetic information. In 1952, Hershey and Chase proved that *DNA* and not proteins carried genetic information encoding molecules, which were then uptaken and transmitted by bacterial cells infected by a bacteriophage. A year later, in 1953, Watson, Crick, and Franklin made the revolutionary discovery of the double-helix structure of *DNA* even though this discovery would take 13 more years before deciphering the genetic code and 25 more years before the first *DNA* sequencing became available.

Margaret Dayhoff was considered the first bioinformatician. Dayhoff recognised the potential of computational resources to solve biomedical problems [1]. From 1958 to 1962, Dayhoff and Robert Ledley developed the first de novo sequence assembler called

COMPROTEIN, created to determine primary protein structure using Edman peptide sequencing data. In the COMPROTEIN software, amino acid sequences were represented in three-letter abbreviations, for ex- ample, Cystine – cys. However, Dayhoff developed the one-letter amino acid code still used today to simplify dealing with protein sequence data. This one-letter amino acid code was used in 1965 on the first biological sequence database: Atlas of Protein Sequence and Structure, created by Dayhoff and Richard Eck [1]. Even if bioinformatics is in the Dayhoff work, the term "Bioinformatics" was only introduced in 1970 by Hesper and Hogeweg to describe the study of informatics processes (computer science) in biotic systems (biology science), becoming a multidisciplinary field (Figure 1) [2].

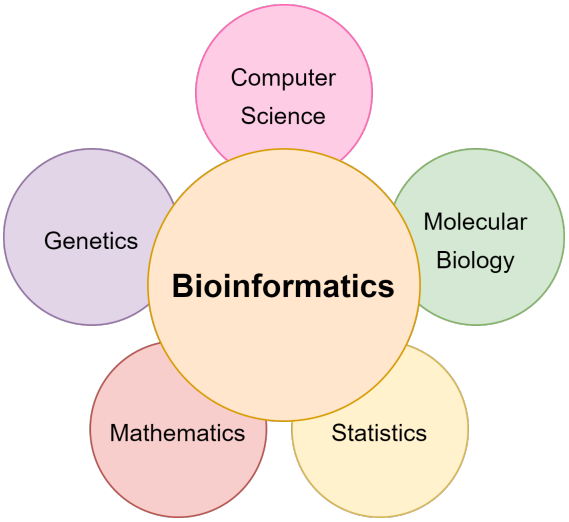


Figure 1: Bioinformatics interdisciplinary field.

Bioinformatics began in the early 1960s with computational methods to analyse proteins sequences, de novo assembly, biological sequence databases, and substitution models.

Posteriorly, *DNA* analysis also emerged due to advances in [2]:

1. **molecular biology methods -** easier manipulation of *DNA* and its sequencing.
2. **computer science -** more powerful computers and new and improved software are used to handle bioinformatics tasks.

In the 1990s through the 2000s, a problem has emerged. The significant improvements in sequencing technology gave rise to an extraordinary data production called "Big Data."

This volume of data can no longer be interpreted by humans alone, leading to new data mining and management challenges, such as analysis, storage organisation, annotation, systematisation, and integration into biological databases, thus requiring additional expertise from computer science into the field [1; 2].

Nowadays, there is a diversity of definitions in the literature and on the world wide web. Here, the description of the NCBI is presented: "*An interdisciplinary field of science in which biology, information technology, and computer science merge together, to allow new biological discoveries and insights and also help create a global perspective from which the principles of Biology can be identified*". The core requirement is computer science, mathematical, and statistical principles, as most biological data analysis tasks are highly repetitive or mathematically complex. However, bioinformatics has several components, usually related to biological molecules, and therefore requires knowledge in different fields like biochemistry, molecular biology, molecular evolution, thermodynamics, biophysics, molecular engineering, and statistical mechanics.

Bioinformatics is not only about modelling or data mining. It is about acknowledging the molecular world. It is a multi-disciplinary field moving from applied to basic science, from developing tools to developing hypotheses. Three of the primary objectives of Bioinformatics include [3]:

1. **Development of new algorithms and statistics** for assessing the relationships among large sets of biological data;
2. **Development of tools** for the analysis and interpretation of the various biological data;
3. **Development of databases** for efficient storage, access, and management of a considerable amount of biological data.

These actions' ultimate purpose is to understand better a living cell and how it functions at the molecular level.

Bioinformatics can provide new insights, a universal perspective, and a better understanding of the cell by analysing molecular sequence and structural data. The conventional flow of genetic information is dictated by the central dogma of biology, where *DNA* is transcribed to RNA, and RNA is translated to proteins, whose capabilities are

ultimately associated with their sequences. Therefore, solving functional problems using bioinformatics has proved to be a productive undertaking [4].

2.1.2 *Databases*

Hıstorical Perspective

A *DB* is a collection of data structured according to a specific database management model, such as: hierarchical, relational, and object-oriented [4]. The most used and popular are *DB*s that employ the relational model. The relational database management model is based on three basic concepts: relationships, attributes, and domains. A relationship is a table with rows and attributes, that is, columns. Each attribute has a specific domain corresponding to the set of values it can take, such as real numbers and alphanumeric characters. A relational *DB* comprises several tables, each table representing a distinct entity (e.g., Patient A, Specialty B, Doctor C) and attributes (e.g., Patient name, Speciality name, Doctor name) of entities. Tables can include [5]:

1. **primary keys -** a set of attributes that uniquely identify each row and cannot be two rows of the table with identical values in all attributes belonging to the primary key. A primary key cannot have a null value;
2. **foreign keys -** a set of attributes projected onto the primary key of another table, thus defining explicit associations between tables.

It is possible to have associations between any attributes of different tables, as long as the respective domains are compatible. Foreign keys only designate explicit associations.

A ***DBMS*** is the software responsible for managing a *DB*, forming a database system. The principal function of the *DBMS* is to allow users and applications to access and/or manipulate data stored in the *DB*, with the *DBMS* being responsible for preserving the consistency of *DB* data [6]. In other others, the operations requested by the different users or applications sent in the context of transactions, that can be one or more operations, the

DBMS will ensure that the execution of each transaction satisfies four properties, also called ACID properties [7]:

- **Atomicity -** ensures that the execution of a transaction. The transaction can be composed of multiple steps that can only succeed or fail:
  - **commit -** the operations included in the transaction is completed executed and committed the changes made are visible;
  - **abort -** the operation failed, and the changes made are not visible.
- **Consistency -** ensures that before and after the transaction the database remains consistent, whether successfully terminated or aborted.
- **Isolation -** ensures that despite the eventual execution of several transactions, each transaction does not see data changed by other transactions that have not been completed and committed to the memory.
- **Durability (Persistence) -** ensures that actions performed by a successful and completed transaction becomes permanent.

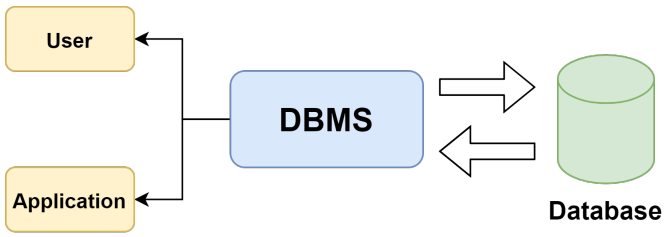


Figure 2: Database Management System.

The SQL is used to access and manipulate data stored in a relational database, and it includes the *DDL* and *DML* [5]. ***DDL* commands** are used to build and modify the structure of tables and other *DB* objects. Also, when they are executed, they take effect immediately, not being included in transactions. There are several *DDL* commands [8], such as:

- *CREATE TABLE* - creates tables;
- *ALTER TABLE* - - changes tables, like changing attributes, adding Foreign Keys, etc.;

- *DROP TABLE* – drops the specified tables.

***DML* commands** allow accessing and manipulating data in tables. These commands can be grouped into transactions and only take effect after the transaction is successful. There are several *DML* commands [8], such as:

- *SELECT* -perform table queries. For this motive, *SELECT* commands are also called queries. These commands return the search results in the form of a table, where the information to be returned as to be specified in the commands;
- *INSERT* - insert data into tables;
- *UPDATE* - actualise data in tables;
- *DELETE* - delete data from tables.

The *SELECT*, *UPDATE* and *DELETE* commands allow specifying the conditions under which the operation must be performed through the *WHERE* clause. This clause specifies a condition, and only if the condition is true will the command be executed. The *WHERE* clause may eventually correspond to the combination of several conditions through other operators, such as *AND*, and *OR*.

It is possible to combine two or more *SELECT* commands, using the *UNION* operator, constructing in this way a table with all the information returned by each command.

Web applications that need to store some information with transactional guarantees resort to relational *DBMS* most of the time, such as Oracle and MySQL support SQL. The integration between web applications and relational *DBMS* is achieved by including SQL commands in the web application code.

Types of Databases

Focusing on Bioinformatics, we can have at least seven types of databases: **(i)** Sequence Databases; **(ii)** Protein Structural Databases; **(iii)** Enzyme Databases; **(iv)** Microarray Databases; **(v)** Pathway Databases; **(vi)** Chemical Databases; **(vii)** Bibliographic Databases.

Sequence Databases are biological databases composed of an extensive collection of automated sequences (*DNA*, *RNA*, protein, or other polymers) where the goal is to store and

distribute this sequence data to the public [9; 10]. The three largest Sequence Databases are: the *NCBI*, the *EMBL*, and *DDBJ*.

Protein Structural Databases are "archives" of structural data of biological macromolecules. The goal of most protein structure databases is to provide integrated structural data resources that correspond to the need of biologists. The database holds deposition and annotation of structural data, providing the bio- logical community access to the integrated resource of high-quality macromolecular structures and related data. The key to structural databases is structural information, while sequence databases are sequence information and contain no structural information for the majority of records. Some of the protein structural databases are the *wwPDB*, *PDBE*, and *PDBj* [11].

Enzyme Databases cover enzyme properties and functions, such as occurrence, structure, or metabolic functions. There are two types of Enzyme databases [12]:

- **Generic Enzyme Databases –** These databases focus on general data about enzymes, such as classification and nomenclature, reaction and specificity, functional parameters, organism-related information, enzyme structure, and Enzyme–disease relationships, to name a few. Examples of these databases include *KEGG* and BRENDA.
- **Specific Enzyme Databases –** Unlike the generic enzyme database, specific enzyme databases are specialised only on certain enzyme classes. Some examples of these databases are MEROPS, *CAzy*, and PeroxiBase.

Microarray Databases contain microarray gene expression data. These databases store, annotate, view, analyse and share microarray data [13]. Some Microarray Databases are *GEO* from *NCBI*, *SMD*, and ArrayExpress from *EMBL-EBI*.

Pathway Databases are a way to associate proteins with their functions and connect them into networks that explain the reaction space of an organism. Several pathway databases have been developed in the past years, such as HumanCyc, *KEGG*, *NCI-PID*, Panther, and Reactome [14].

Chemical Databases are a powerful tool for discovering new lead compounds. The growth of chemical databases in drug discovery is one of the key points of these databases.

Using a query that captures the idea of the requirements for biological activity, other untested compounds with identical types of activity can be identified. Most of the compounds found in a search will not act as the new drug itself, but it could indicate novel ways to provide the desired activity giving the researchers a new idea [15]. Some of these databases are PubChem, ChEMBL, and ZINC.

Bibliographic Databases are defined as "*digital collections of references to published sources, particularly to journal articles and conference proceedings, which are tagged with specific titles, author names, affiliations, abstracts, and identifiers*". The PubMed and *MEDLINE* are well- known bibliographic databases [16].
All the types of databases mentioned before have specific mechanisms to obtain the type of item that is required, such as keywords to retrieve the information associated with it, making the user's experience more accessible. It's also important to mention that some databases require subscriptions, and some are free to all users.

2.2 TEXT MINING

In his book "Predictive Analytics", Eric Siegel provides an analogy: "*if all the data in the world was equivalent to the water on earth, then textual data is like the ocean, making up a majority of the volume*" [17]. This analogy is interesting because currently, there is a large amount of literature in unstructured text, making unstructured text the new frontier of data science. What for humans can be a straightforward task for computers can be arduous [18]. Unlike numeric or categorical data, natural language is challenging for computers because it does not exist in a structured format consisting of rows (of examples) and columns (of attributes). Given the volume of available documents and still growing, researchers have been using computers' capabilities, such as data processing and storage, to hasten the process [19]. However, as mentioned before, the interpretation of the documents is a complex task for computers, so this brings the need for tools that can reduce the time and cost of curation, increase annotation consistency, and provide the relations to support evidence in the literature, making helpful annotations to researchers, through *TM*. Some of the first *TM* applications came about when people attempted to organise documents.

Contrary to popular belief, *TM* is different from a simple web search. In a web search, the user is seeking something already known and has been written by someone else. The purpose is to find content, but *TM* goes further into analysing the text, extracting useful information, not solely finding and retrieving documents. But the real problem is retrieving only the material with relevant information to the user [18]. Thus, *TM* also known as Intelligent Text Analysis, Text Data Mining or *KDT*, are the technologies and tools used to process, analyse and find relevant and non-trivial information in the unstructured and semi-structured text. *TM* is an interdisciplinary field specialising in information retrieval, *ML*, data mining, computational linguistics, databases, and statistics, as shown in Figure 3 [20].

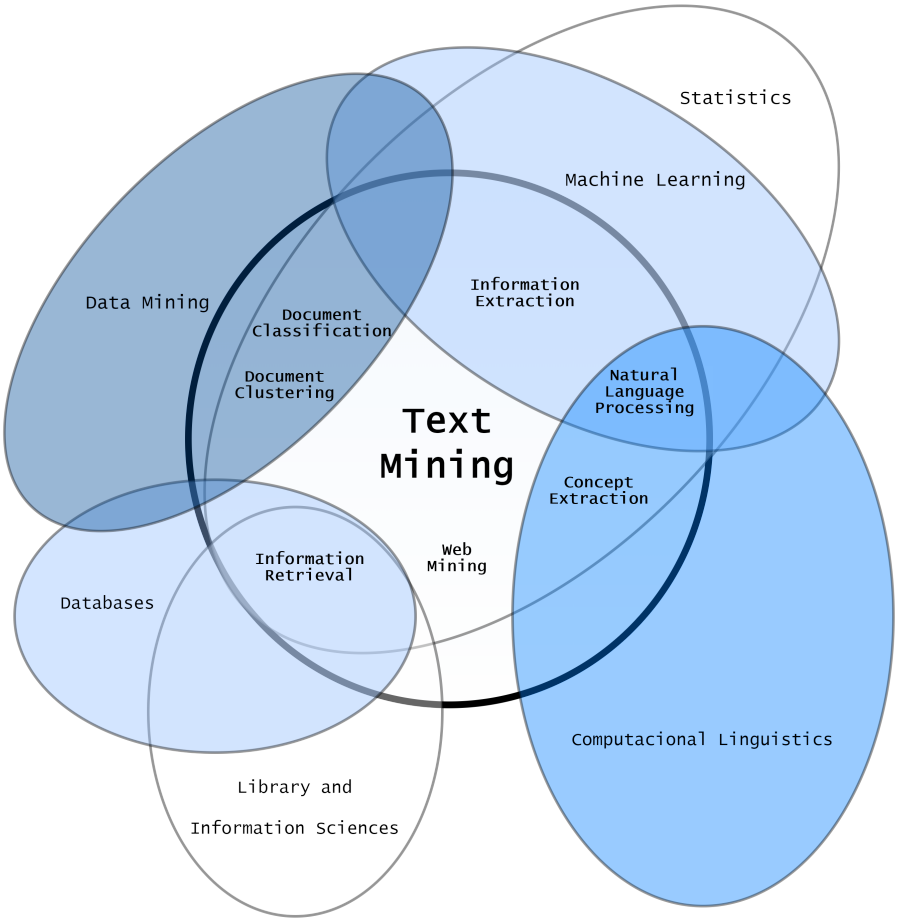


Figure 3: Venn diagram representing the interdisciplinary field of text mining. Adapted from [20].

TM is applied to transform information in a structured form suitable for analysis or drive *ML* algorithms. It has several accomplishments such as annotating databases, sharing

knowledge between fields, and supporting corporations with decision-making (operational and strategic) [21].

As mentioned before, *TM* uses different technologies and tools, making the sub-division of *TM* unclear. Therefore, the following points will follow the division by Miner, G. et al., which separates *TM* into seven distinct groups [20]:

1. ***IR*** can be defined as software that deals with storage, organisation, evaluation, and retrieval of information from document databases, predominantly textual information. The term *IR* was coined by Mooers (1951), who established that *IR* is the ability of the user to express the required information and systems for searching and returning the information. Thus, *IR* is the action of obtaining information, namely, text, images, sounds, and even video sequences, satisfying the users' needs from the sets of data stored on databases. However, *IR* does not unequivocally return the answers the user wants; it only processes a collection of documents stored in the databases. *IR* is also known as "search", which has been described as an "integral application" in the modern days. Some examples of the commonly used search engines are Google and Yahoo[20; 22].

2. **Document Clustering:** uses data mining clustering methods to group documents, terms, para- graphs, or snippets. Document clustering can be useful in *IR* tasks, such as information retrieval by search engines. Clustering algorithms, such as k-means or Hierarchical clustering, group documents into clusters using the intra-group similarities and the low inter-group similarities. The number of clusters will depend on the method used [20].

3. **Document Classification:** This technique uses different classification methods to group and categorise documents, according to models trained on labelled examples. This will be explained in more detail in subsection 2.2.1, as it plays an essential role in the present work.

4. **Web Mining:** Nowadays, the amount of text-based data emerged on the WWW in a freeway for user's access. Therefore, the attention of the researchers is on data present

on the Web, making the need for data mining techniques that focus on the Web, giving rise to Web Mining [20; 23].

5. **IE:** Is the process of transforming unstructured text to structured data by identifying and extracting information such as entities, relationships between entities, and attributes describing entities. It requires more profound analysis than keyword searches; therefore, it is not straightforward [24; 25].

6. ***NLP*:** Is an area of research and application that investigates how computers analyse and manipulate natural language. The goal is to collect knowledge on how humans understand and use language, and develop proper tools and techniques, allowing computer systems to understand natural language and perform various tasks or applications. *NLP* tools will be explained in more detail in subsection 2.2.2, and *NER* in subsection 2.2.3 .

7. **Concept Extraction:** Is the technique of extracting the concepts of a document. It analyses words (a single word or a multi-word term) and groups them by similarity. The similarity of the words is measured by modeling a statistical features (e.g., frequency of occurrence) and linguistic features (e.g., part-of-speech) [20].

However, the tasks mentioned above are usually case-specific. Hence, existing mining tools for generic texts may not work or fail when applied to the biological context. As biological data is particular and complex, a collection of freely available text mining tools adapted to the needs of the field have been developed. Thus, a sub-field of text mining named *BioTM* has emerged. *BioTM* may have begun with the first paper published on the medical domain decades ago, almost 30 years since the first paper on *TM* in the genomics domain [26]. *BioTM* has several objectives depending on the literature topic (biology, Chemistry, Patient Records, etc.) [27]. Here one of the goals of *BioTM*, which is identifying and extracting information on biological entities (genes and proteins), phenotypes, or biological pathways is presented. Moreover, *BioTM* helps manual curating biological databases [28]. Examples of *BioTM* tools and platforms developed for the community are @Note [29] and *BioDR* [30].

Therefore, *BioTM* techniques improved access to knowledge to a large audience of biologists, obtaining results easier and faster.

2.2.1 *Document Classification*

Document Classification is sometimes called Text Classification or Text Categorization. Text Classification presents textual data containing a phrase, a sentence, or a complete document with paragraphs. In Document Classification, the term "document" can cover a lot of formats such as writing, videos, drawings, or presentations. Hence, the term "document" covers the textual data that Text Classification refers to. In Document Classification, the objective is to classify documents having the same essence as Text Categorization, despite using classification or a supervised machine learning approach to classify or categorise text [31].

Document Classification can be defined as the process of assigning a predefined set of labels to a document [20; 31; 32]. This process can be divided into two classifications problems [32; 33]:

- **binary classification -** two labels;
- **multi-classification -** more than two possible labels.

Document Classification has four different levels [34]:

- **document level –** the algorithm obtains the relevant categories of an entire document;
- **paragraph level -** the algorithm obtains the relevant categories of a single paragraph;
- **sentence level -** the algorithm obtains the relevant categories of a single sentence;
- **sub-sentence level -** the algorithm obtains the relevant categories of sub-expressions in a sentence.

Document Classification has broad applications, such as news, music, or movie categorisation, document sorting, text filtering and indexing, sentiment analysis, language detection and spam filtering, through algorithms that analyse characteristics/features of a document (title, length, words used, file name, or *URL*) [32]. For example, spam filtering analyses an incoming email and determinates if it is spam or not (this is a binary classification with two possible labels: "spam" or "not spam"), with a simple features

analysis, but there are also solid indicators of spam, such as the dollar sign ($) or the number of exclamation points used (!!!!) on the email [20; 31; 32; 34; 35; 36].

In the early days, document classification was performed manually by specialists familiar with the documents' topics, being very accurate. However, the number of documents to be classified increased exponentially, creating scalability problems [31; 36]. Recently, techniques to improve efficiency and hasten this process, such as employing *ML*, have been developed. There are two main types of *ML* approaches relevant to document classification, i.e., unsupervised and supervised *ML* [31]. Unsupervised and supervised learning algorithms will be detailed in subsections 2.3.1 and 2.3.2.

In conclusion, classification can solve various problems and automate otherwise time-consuming operations [31].

2.2.2 *Preprocessing*

Preprocessing has an essential role in *TM* techniques and applications [37; 38]. *ML* models operate entirely with numerical vectors; Thus, text data (words and phrases in unstructured information) must be converted into numerical vectors. This process is called vectorization. For this to be possible, preprocessing is usually carried out. There are several preprocessing techniques, such as:

1. Filtering – it follows a set of rules for eliminating duplicate strings and irrelevant text [37];
2. Splitting Sentences – used to break text into sentences. This break into a sentence can be something simple as splitting the string on (.) [39];
3. Tokenization – splitting sentences of textual content into small meaningful elements called tokens (words, characters, or symbols). The complexity of these tasks varies according to the needs of *NLP* and the complexity of the language [37; 39; 40];
4. Stemming – rule-based process to reduce inflectional or sometimes derivationally related token forms to a common base form (word stem, root, or base) [38; 39; 41; 42];

5. Stop Words Removal - these words are considered to have little or no importance in *TM*, as they make the corpus heavier. Thus, words commonly found in the corpus, such as articles, prepositions, and pronouns, do not contribute to the meaning of the documents. Examples of stop words are: ”the”, ”in” and ”a”, to name a few [37; 39; 41; 43; 44].

NLP enables computers to process human language data through the combination of computational linguistics and statistical *ML* models. Several *NLP* tools are available, such as the famous Python toolkit *NLTK*. *NLTK* is an important platform that provides interfaces to work with over 50 corpora (large and structured set of texts) and lexical resources, alongside text processing libraries like those mentioned above [39; 43; 45].

Regarding vectorization, the goal is to decrease the data dimension, by converting them into significant numerical representations. One of the most used vectorization techniques in *ML* is the Bag-of-Words approach that allows representing text as numerical feature vectors. The idea behind the bag-of-words model counts how often each word occurs in a sentence or a particular document, e.g., the antibiotic adjuvants can save lives → adjuvants, antibiotics, can, lives, save, the. There are two reasons for this. The first is that this representation is computationally easy to work, and the second is that this representation breaks drown the text into fragments of meaning and serves as a way of capturing the text's meaning [46].

2.2.3 *Named Entity Recognition*

Named Entity Recognition, also called *NER*, is a natural language processing technique. The goal of *NER* is to identify all mentions of the named entities in the text. *NE* are noun phrases that refer to specific types of entities (e.g., organisations, persons, location, etc.) [39; 47]. *NER* can be divided into two tasks, identifying *NE* in a text and classifying them into categories, such as proper names, places (e.g., cities, countries, etc.), expressions, etc. *NER* allows extracting essential information to understand a text or merely collect necessary information and store it in a database. This context is used to retrieve different entities such as bacteria, antibiotics, adjuvants.

NER has three distinct techniques:

1. **dictionary-based:** uses dictionaries or lexicons of entities to match entities of interest with the total document words [48];
2. **rule-based:** uses the knowledge that can take the form of [48; 49]:
   a) general knowledge about how language is structured;
   b) specific knowledge about biological entities in the literature, and the variant forms by which they might be mentioned in the literature;
   c) any subset or combination of these, thus matching entities of interest;
3. **machine learning-based approaches:** train the system through examples with associated features, then use mathematical models to predict the possible features of an unannotated dataset [48; 49].

BioTM has several tools and platforms developed for the community to perform this task, such as the tools mentioned before:

- **@Note** is a set of user-friendly tools for biomedical document retrieval, annotation, and curation [29];
- ***BioDR*** allows the semantic indexing of the query's results by identifying relevant terms in the documents [30].

2.3 MACHINE LEARNING

ML is a subfield of AI to develop computer algorithms that can identify and recognise non-trivial patterns and underlying connections within the different datasets, allowing the creation of classifiers for novel data. *ML* can be applied to various fields, such as biology, geology, politics, finances. Hence, any problem and field that needs to interpret and act on data [50; 51; 52]. Suppose you want to buy something and use a search engine that shows multiple links with relevant content. When opening the links, the search engine will learn. The spam folder in your email account holds unwanted emails through filtering approaches

that detect and filter these emails. Companies use *ML* to improve business by helping make decisions, detecting disease, forecast weather [53; 54]. The algorithms are the machine "intelligence" backbone with their distinctive ability to learn from the input data [50].

ML is generally classified into four groups: Supervised Learning, Unsupervised Learning, Semi-Supervised Learning, and Reinforcement Learning [50; 52].

- **Unsupervised Learning** uses unlabeled data. There is no correct answer or a supervisor. Thus, the algorithm uses the learned features to identify the class when new data is introduced.
- **Supervised Learning** aims to map an input to an output through a function based on features and patterns associated with labelled training data.
- **Semi-Supervised Learning** combines supervised and unsupervised Learning, using labelled and unlabeled data. The goal is to classify the unlabeled data through the labelled data.
- **Reinforcement Learning** takes actions in the surrounding environment to maximise some notion of reward, being these the final goal of the models.

2.3.1 *Unsupervised Learning*

Unsupervised learning models are trained using unlabeled data. The machine receives a set of data (or inputs), such as an image or words in a document, without supervised target outputs or rewards from its environment to predict the output. The algorithm must learn and create new labels/classes from which it has learned from the data through self-learning, where the algorithm can find previously unknown patterns in datasets. In short, the central goal of unsupervised Learning is to explore the relationships within the given observations and group them into similar clusters [55; 56; 57]. Imagine this scenario, your friends invited you to watch a volleyball match, but you do not know anything about that sport. When the game starts, you are blank, but you begin to learn how the game works. Analysing what you see, you can obtain certain conclusions, such as there are two teams, different kinds of

players (libero, setter, etc.), the game has at least three sets that end at 25 points but with the addition of having at least 2 points difference from the other team and teams switch camp every set. Thus, you have all material, and you learned volleyball all by yourself without supervision.

Two examples of classic unsupervised Learning are clustering (e.g., Document Clustering) and dimensionality reduction (e.g., *PCA* and *t-SNE*). The clustering goal is to find similarities and differences and dimensionality reduction to find a less complex representation of the data. The commonly used algorithms in unsupervised Learning are hierarchical clustering, which retrieves more detailed information, and K-means clustering, being more efficient than hierarchical clustering.

2.3.2 *Supervised Learning*

Supervised Learning models are trained using labelled data. That is, prior knowledge about the behavior of the dataset is present, where each input has a specific label attributed. The goal of Supervised Learning is to find a way to establish the relationship between the inputs (features) and the output in the best way possible. First, the input dataset is split into train and test sets. In a first instance, the algorithm will receive the train set, a known set of input data, and its output to learn patterns. The training process will continue, and when the level of performance is high enough, the algorithm should predict the output of the test set. There are two types of outcomes in Supervised Learning: the target output initially assigned to each training example and the machine-generated output by the algorithm [50; 52; 57; 58].

Supervised Learning uses classification algorithms to map the input to output labels or regression techniques to map the input to continuous output. There are numerous Supervised Machine Learning algorithms, such as linear regression, logistic regression, naive Bayes, *SVM*, k-nearest neighbors, decision trees, and several others [50].

2.3.3 *Reinforcement Learning*

Reinforcement learning implies an existing problem that an autonomous agent faces. The agent has to learn behavior through trial and error interactions with the surrounding environment [50; 52; 59].

The goal of reinforcement learning is to maximise a reward through exploitation and exploration. To conquer a reward, the agent must prefer actions that it has tried in the past in the environment, and knows to be efficient in producing a reward. However, it has to try approaches different from those taken previously to perform such actions. The agent has to exploit what it has already experienced to obtain a reward, but it also has to explore to obtain more data and improve the actions. The exploration and exploitation take to failure, making the agent go through several actions and gradually favor the ones that seem to be best. All the reinforcement learning agents have specific goals, can sense characteristics of the environment, and can select actions to influence their environments. Examples of reinforcement learning can be a computer playing chess with the human, self-driving cars, and learning to recognise spoken words [52; 60].

2.3.4 *Evaluation of Machine Learning Models*

The classifiers' performance in Supervised Learning is evaluated by comparing the actual and predicted labels. The predicted labels from the classifiers are commonly called positive or negative. These labels are called by their notation [61]:

- ***TP* -** when the classifier correctly predicts a positive label;
- ***FP* (Type I error) -** when the classifier incorrectly predicts a positive label;
- ***TN* -** when the classifier correctly predicts a negative label;
- ***FN* (Type II error) -** when the classifier incorrectly predicts a negative label.

These notations allow computing different metrics, such as Accuracy, Recall, Precision, and F1-Score, ranging between 0 and 1. Below are the equations of every metric mentioned.

$$\textbf{Accuracy} = TP + TN \,/\, (TP + FP + FN + TN)$$

$$\textbf{Recall} = TP \,/\, (TP + FN)$$

$$\textbf{Precision} = TP \,/\, (TP + FP)$$

$$\textbf{F1-Score} = 2 \times (\text{Precision} \times \text{Recall} / \text{Precision} + \text{Recall})$$

Accuracy is the exact match of the labels predicted that correspond to the actual labels. **Recall** (aka True Positive Rate or Sensitivity) is the ability of the classifier to find all the positive items. On the other hand, **Precision** is the ability of the classifier not to label as positive an item that is negative. The **F1-Score** can be interpreted as a weighted average of the Precision and Recall.

Commonly, the metrics above can complement these metrics in binary classification with ***ROC*** curves, ***PR*** curves, and the respective ***AUC***, which is a summary of the information in the curve [62; 63]. *ROC* plots the proportion of correctly labelled items, - Recall, against the proportion of incorrectly labelled items - *FPR*, while the *PR* plots Precision against Recall [62; 64].

$$TPR = TP \,/\, (TP + FN)$$

$$FPR = FP \,/\, FP + TN = 1 - specificity$$

The dataset balance is of paramount relevance for the interpretation of these metrics. When the dataset is unbalanced, it contains few elements in one class, such as relevant documents (positive documents). Thus, in this scenario, *ROC* curves should not be used, as they regard *TP* and *FP* and do not consider the *FN*, providing biased results. For this motive, *PR* curves are a helpful alternative for the unbalanced datasets because they highlight performance differences lost in *ROC* curves [62]. The *AUC* measure is the capacity of a classifier to distinguish between classes, helping us visualise how well our classifier is performing, being commonly used to compare distinctive classifiers. The *AUC* measure ranges in value from 0 to 1, a classifier whose predictions are best to have values close to 1 or even 1.

2.3.5 *Feature Generation & Feature Selection*

The data inputted into an *ML* algorithm is called "features." Features are the basic building blocks of datasets, and the quality of the features significantly impacts the quality of the insights obtained when using *ML* models.

Feature generation creates new features from existing features, making further information more accessible during the model construction and hopefully resulting in a more accurate classifier. Feature generation is also known as feature extraction, feature construction, or feature engineering. However, there are different interpretations of these features. The most common feature generation approach is the Bag-of-Word approach, in which a document is represented as a vector of length n (number of terms in the paper). However, this approach has the disadvantage of ignoring semantic correlations between terms [65].

Feature selection is a process that takes a subset from the original set, obeying criteria. Its purpose includes reducing dimensionality by removing features that are loosely related to the output that might affect the learning process by decreasing the effect of the important ones. Feature Selection is also used to remove features strongly coupled with other features that do not provide extra information and can make the result wrong. These features can further lower training performance by straining computational resources such as time and memory. This subset has the same meaning as the original one, but it better understands data and Learning. After this process, the algorithm can be applied for text classification [65; 66].

2.4 WEB FRAMEWORK

Web Application Framework or simply Web Framework provides tools and libraries that do common web operations to simplify the development of the application.

At the beginning of web development, the application was hand-coded. However, in 1995 problems connected with changing the structure of an application were resolved. With this, web-specific languages emerged as well. Web frameworks can be monolithic or modular. A monolithic framework provides a tightly coupled codebase (every part of the code makes assumptions about the other part), usually including all the functionalities required by a web application. On the other hand, the modular framework is loosely coupled (every part of the code communicates with other components through more-or-less standardised and neutral interfaces), providing the minimum of functionalities and structure.

Frameworks have two groups for development, the client-side (the back-end) and the server- side (the client-side). The front-end is, also known as the user side, is what the user interacts. The front-end builds the *UI* (the buttons, inputs, layouts, images, etc.) and the *UX* (loading time of the website, the chatbox, etc.). The back-end comprises the server and the database. Servers control how users access files and databases are organised and structured data collection. Servers control how users access files and databases are organised and structured data collection. Some examples of Front-end and Back-end frameworks are:

- **Server-side:**
  - **Django [67]:** Python Monolithic framework.
  - **Express [68]:** JavaScript Modular framework.
  - **Flask [69]:** Python Modular framework.
- **Client-side:**
  - **React [70]:** JavaScript Modular framework.
  - **Angular [71]:** JavaScript Monolithic framework.
  - **jQuery [72]:** JavaScript Modular framework.

2.4.1 *Django and Django REST Framework*

2.4.1.1 *Django*

As mentioned before, Django is a free and open-source framework, that accelerates the development of websites, created by Adrian Holovaty and Simon Willison in 2003. This framework was built to handle the difficult part which is the creation of database-driven websites, concentrating on less code, reusable components, and rapid development. Django is characterized as:

- **Fast -** releasing the developers from reinventing the wheel and giving them the ability to focus on creating the application [67].
- **Secure -** considers the most common security mistakes, such as *SQL* Injection and *XSS*, helping developers to avoid them [67].
- **Scalable -** uses a "shared-nothing" architecture, meaning that each layer is independent, which guarantees scalability at any level [67].
- **Versatile -** can be used for a variety of purposes. Governments, organizations, and companies have already used this framework [67], including NASA, Spotify, and Reddit [67].
- **Fully Loaded -** provides extras that can be used to handle common web development tasks, such as the creation of models, authentication, the design of the *URL*'s , writing views, etc [67].

Django uses **Models** to manage and access data, defining the structure of the stored data, such as field types and respective behavior. A model usually maps to a single database table, where the definition of the model is independent of the database, as Django will handle the communication with the database after writing the model structure [73].

Models are defined in an application *models.py* file, being implemented as subclasses of django.db.models.Model, including multiple fields (corresponding to a database column), methods, and metadata. Figure 4 presents a model with two fields (title and abstract), of

type *models.CharField*, meaning that these fields will have strings of alphanumeric characters. These field types have arguments that specify how the field can be used or stored.

```
from django.db import models


    class Article(models.Model):
        title = models.CharField(max_length=30)
        abstarct = models.CharField(max_length=30)
```

Figure 4: Example model defines an Article, which as a title and abstract fields.

Views are methods that receive a request on a specific *URL* and return the respective response. The response can be an *HTML* template response, redirecting to another page, an image, a 404 error, etc. The view itself contains the necessary logic to return that response [74].

All the view functions necessary for the application are defined in a views.py file. Figure 5 presents a view function that takes a HttpRquest object as a parameter named request and returns an HttpResponse object with the generated response.

```
from django.http import HttpResponse
import datetime

def current_datetime(request):
    now = datetime.datetime.now()
    html = "<html><body>It is now %s.</body></html>" % now
    return HttpResponse(html)
```

Figure 5: Example of a view that returns current date and time.

The ***URL*s module** is mapps the *URL* endpoint to the respective view function. In Django, to design this *URL* for an app, it is necessary to create a python module URLconf (*URL* configuration). URLconf is a collection of patterns that Django will match against the requested *URL* to find the specific view [75] [76].

```
from django.urls import path

 from . import views

 urlpatterns = [
     path('datetime/<str:datetime>/', views.current_datetime),
 ]
```

Figure 6: Example of a URLconf.

Figure 6 presents a URLconf. When a user makes a request, this will be the algorithm the system will follow to definitive which code to execute [76]:

- **Determinate the root URLconf module to use.** This information will be present in the setting.py file in the variable ROOT_URLCONF.
- The **URLconf model will be loaded and will look for the variable urlpatterns.**
- **Each *URL* pattern will be matched against the requested *URL*** in order, stopping with the first one that matches the path_info (in this case will be the <str:datetime>).
- After the *URL* patterns matches, **Django will import and call the specific view**.
- If no *URL* pattern matches or an exception is raised, Django invokes an error-handling view.

2.4.1.2 *Django REST Framework*

DRF is an open-source toolkit for developing a web *API*. This framework is a library using the Django structure to build a RESTful *API*, where it is possible to transfer information between an interface and a database easily. *DRF* presents some benefits compared with writing resources with Django packages:

- **Flexible and Refined serializers -** *DRF* serializers work with complex data strucutre and prevent serialization errors [77].

- **Integrated authentication mechanism -** offers the developer the restriction of access to resources and saves time. Some of the authentication mechanisms that *DRF* offers are *JSON* Web Token, and OAuth2 [77].
- **Browsable *API* -** provides a user-friendly output of the resources through the web browser [77].

The *DRF API* is composed of 3 layers:

- **Serializers –** are one of the main building blocks of *DRF* that converts the data from the server and serializes it into a format that can easily be transferred through an *API* and be read by other technologies [77] [78].
- **Viewset –** is a group of views which that defines a set of actions - like list, create, and retrieve- making the code consistent and not repetitive [77] [78].
- **Router -** specifies the *URL*s that will give access to each viewset [77] [78].

The utilization of the *DRF* requires specific knowledge. Therefore, some definitions will be presented in the Appendices, section A.1, including the definition of *API*s and *REST*.

2.4.2 *React*

React is often mistaken for a framework [79], but it is actually an open-source JavaScript library for building user interfaces. React uses its own syntax. *JSX* is a JavaScript Extension that provides an *HTML*-like syntax extending ECMAScript so *HTML* syntax can co-exist with JavaScript code [70].

In React the *DOM* updates, event handling, and attribute manipulation are not made with the browser *DOM* but with the *VDOM*. *VDOM* is considered a pattern where a virtual representation of a user interface is maintained in memory and synced with the actual *DOM*, improving the efficiency and speed of rendering [70]. React applications are built using reusable components. **Components** are reusable blocks of code that divide the user interface into smaller pieces. A component has its own *HTML*, and *JS* file that can be used anywhere in the application. In a component-based webpage, each section is divided into

self-contained pieces of interface like the search bar, headers, and main content, where all are built as separated components and each of them will have their code. React has two ways of writing components, namely the function-based and class-based [70].

Class Components extends from *React.Component* being written as JavaScript ES6 classes. Class components were the only way to include lifecycle and state methods features in React, being considered stateful. They can encompass variables, functions, props, and states with *this* reserved keyword. Figure 7, presents a class-based component [80].

```
class Welcome extends Component {
  render() {
    return (
      <h1> Hello {this.props.name}!  </h1>
  }
}
```

Figure 7: Example of a Class Component.

Function Components (or functional components) are simple JavaScript functions that can optionally accept data as a parameter. They are described as lighter than class-based components since are less complex and use less code as shown in figure 8 [80].

```
function Hello(props) {
   return <h1> Hello {props.name}!  </h1>;
}
```

Figure 8: Example of a Function Component.

Function components were stateless until the introduction of Hooks in React 16.8, which allowed achieving most of the ability of class components such as state and lifecycle methods [81].

React Hooks promotes cleaner code, which can be simply maintained and reused. This approach provides a chance to implement all the logic directly inside function components, like a basic operation with state [81].

Although Hooks are similar to JavaScript functions there are two rules to follow:

- **Only call Hooks at the top level -** Hooks should not be called inside conditions, loops, or nested functions. Instead, they should be used at the top level of React function to guarantee that they are called in the correct order each time a component renders, allowing to preserve Hooks between multiple calls [81].
- **Only Call Hooks from React Functions -** Hooks should be called from custom Hooks or React function components, not regular JavaScript functions. This rule ensures that all stateful logic in a component is visible from its source code [81].

2.5 ANTIBIOTIC RESISTANCE

Antibiotics are one of the most significant discoveries for health. They are distributed into different classes and each class of antibiotics favors a specific mode of action, with three primary targets in bacterial cells: cell wall synthesis, protein synthesis, or nucleic acid synthesis. Briefly, each mechanism of action will be described [82]:

1. **Inhibition of Cell Wall Biosynthesis -** Bacterial cells are surrounded by a cell wall made of peptidoglycan (long sugar polymers). By the action of transglycosidases, the peptidoglycan undergoes cross-linking of the glycan strands, and the peptide chains extend and form cross-links, one peptide to another. In the presence of PBPs, the D-alanyl-alanine portion of the peptide chain is cross-linked by glycine residues. This cross-linking strengthens the cell wall lactams and the glycopeptides inhibit cell wall synthesis.
2. **Inhibition of Protein Synthesis -** First, there is a process called transcription, where the information retained in the bacterial *DNA* is used to synthesise an *RNA* molecule referred to as the *mRNA*, then the translation takes place, where the ribosome synthesises proteins present in the *mRNA*. Ribosomes and cytoplasmic factors perform protein biosynthesis. The 70S bacterial ribosome is constituted by two subunits: 30S and 50S. Targeting the 30S and 50S subunit of the bacterial ribosome, the antibiotic can inhibit protein biosynthesis with antimicrobials.

3. **Inhibition of *DNA* Replication -** One of the significant targets of antibiotics is nucleic acid replication and repair mechanisms. The class of synthetic antibiotics of quinolones targets *DNA* gyrase (*DNA* topoisomerase II), which helps unwind replicated *DNA*, making cuts in both strands of *DNA*, thus relieving torsional stress in the replanting supercoil. The quinolones bind to these cleavage sites, thus stabilising the *DNA* and preventing further system repair.

4. **Inhibition of Folate Biosynthesis -**Sulfa drugs and trimethoprim inhibit distinct steps in folic acid metabolism. A combination of sulfa drugs and trimethoprim shows synergy and a reduced mutation rate for resistance. The sulfonamides competitively inhibit dihydropteroate synthase with a higher affinity for the enzyme than the natural substrate (p-aminobenzoic acid), and agents such as trimethoprim act at a later stage of folic acid synthesis, inhibiting the enzyme dihydrofolate reductase.

However, these mechanisms of action have not prevented from reaching an era in which bacteria are resistant to antibiotics. The theme "antibiotics resistance" is not 'new'. Since early 1945, Sir Alexander Fleming raised the alarm regarding antibiotic overuse, and studies about this issue have demonstrated a direct relationship between antibiotic consumption and the dissemination of resistant bacteria strains. Moreover, bacterial infections have once again become life-threatening due to the lack of efficacy of antibiotics. Besides overuse, there are other factors such as inappropriate prescribing, extensive agricultural use, and the lack of new antibiotics. These factors take us to an era where antibiotics are no longer so effective [83].

This resistance is achievable because bacteria have distinct mechanisms to resist the antibiotics, such as:

1. **Efflux pumps -** they are found in Gram-positive and Gram-negative bacteria, and actively transport small molecules out of the bacterial cell. These pumps are classified into five families, depending on their structure, species distribution, substrate specificity, and energy source. Some of these pumps remove various substrates, which contributes to multidrug resistance [82; 84].

2. **Impermeability of the membrane -** The low permeability of the outer membrane in gram-negative bacteria contributes to resistance of antibiotics. This outer membrane or "outer membrane barrier" appears to be an extremely sophisticated macromolecular assemblage that performs the crucial role of providing an extra layer of protection. The low permeability of the outer membrane is a challenge to antibiotics because it requires increasingly larger doses of antibiotics for being effective [85; 86].

3. **Modification of the target -** Mutation or modification of the binding sites of the antibiotics targets causes rapid resistance because it directly affects the target structure. These modifications can reduce susceptibility to inhibition of the antibiotic, but the cellular functions of the target are retained. Due to changing the target, the cell adapts to compensate for new characteristics of the modified target. In conclusion, the slightest alteration of the target will change the binding affinity of antibiotics [82; 84; 85].

4. **Degradation or inactivation of the antibiotics -** This strategy relies on chemical transformations to lower the effect of the antibiotics. This effect can be fought through hydrolysis reactions, group transfers, and redox mechanisms, by changing the enzymes' ability to degrade or inactivate the antibiotic's structural key elements. One of the most sensitive antibiotics to inactivation is the lactam because pathogens have evolved a lactamases enzyme to cleave the critical lactam ring [82; 84].

5. **Genetics of Resistance –** Many studies have revealed the presence of genetic loci involved in the resistance. There are two types of resistance **(a) natural resistance -** specific property of the bacterium as a result of functional and intrinsic characteristics and **(b) acquired resistance -** mutation of the bacterial genes or transferable resistance to the next generation through cell division and shared among bacteria through horizontal gene transfer [84; 87].

These antibiotics' resistance mechanisms are passed to the next generation through cell division and shared among bacteria through horizontal gene transfer.

2.5.1 *The role of biofilms in antibiotic resistance*

Regardless of the variety of antibiotic resistance mechanisms, microbial persistence is attributed mainly to biofilms because biofilm cells can be 10-1000-fold more resistant to antibiotics than free-living microorganisms. Biofilms are a heterogeneous congregation of surface-associated microorganisms encased in a self-synthesised polymeric matrix consisting of polysaccharides, lipids, protein, and *DNA*, growing attached to a biotic or abiotic surface. Moreover, biofilms are coordinated functional communities where microorganisms can "communicate" to each other and respond using chemical signal molecules. Biofilms are ubiquitous and thus they can be present in almost every environment condition (e.g. nature, industry, and hospitals).

Infections associated with biofilms include cystic fibrosis, infectious kidney stones, otitis media, infective endocarditis, chronic inflammatory diseases, etc [88]. It is estimated that biofilms cause 80% of microbial infections [89], and their notable tolerance to several classes of antibiotics leads to chronic infections.

Biofilms are formed through five stages of development: **(i)** initial attachment, **(ii)** irreversible attachment, **(iii)** maturation, and **(iv)** dispersion. . The first stage starts with the attachment of free-floating cells to a solid surface.

1. At the **first stage**, microorganisms attach to a solid surface and further microorganisms use cell adhesion structures, e.g. pilli, to attach tightly to the surface. This attachment is reversible, as bacteria can dissociate and return to the planktonic form [90; 91].

2. In the **second stage**, the bacteria communicate through signals and once the signal intensity reaches a threshold, bacteria start to produce a matrix of extracellular polymeric substances which makes the attachment irreversible [90; 91; 92].

3. The **third stage** occurs the maturation of biofilms, where cell-to-cell communication or quorum sensing plays a vital role. At this stage, microbial cells secrete small chemical signal molecules, and when the amount of these molecules reaches a threshold concentration, they induce the regulation of a panoply of genes, including those associated with virulence factors, crucial for infection development [90; 91].

4. The **fourth stage**, is the dispersion or detachment stage, where cells detach from microbial biofilms to seek new areas to form new biofilms [90; 91].

Several factors can accelerate biofilm formation, such as an adequate colonisation area, poor nutritional conditions, environmental stresses (e.g. altered pH, antimicrobial action, host immune action).

The biofilms are very resistant to antimicrobial agents, since the mechanisms of protection working on biofilms appear to be distinctive from those responsible for traditional antibiotic resistance [93]. The traditional definition of antibiotic resistance is a rise in the MIC of an antibiotic due to a change in the cells, such as horizontal gene transfer and acquisition of resistance, among others. However, this definition is not so "linear" when applied to biofilms. In biofilms, the resistance mechanism prevents the antibiotic from accessing its target and the tolerance mechanism shuts down the targets of antibiotics [94].

Mechanisms of resistance in the biofilm include **(i)** slowly penetration of the antimicrobial agent through the matrix, **(ii)** cellular slow growth, **(iii)** persister cells, **(iv)** adaptative responses, **(v)** phenotypic heterogeneity, and **(vi)** quorum sensing. Each mechanism will be briefly described:

1. **Slowly penetration of the antimicrobial agent–** The antimicrobial molecules may fail to penetrate the biofilm's surface layer because of slow penetration. These molecules must diffuse through the biofilm matrix to inactivate the encased cells. If the molecules penetrate slowly into the biofilm, the enzymes (e.g. beta-lactamases and chloramphenicol acetyltransferase) produced by the biofilm cells will enter into action and inactivate or modify the antibiotic. Moreover, the biofilm matrix exhibit properties identical to a diffusional barrier for the antimicrobial molecules, influencing the velocity of transport to the interior of the biofilm or the interaction of the antimicrobial constituents and the matrix constituents [93; 95; 96; 97].

2. **Cellular slow growth –** Biofilms include slow-growing or stationary phase cells, making the uptake of antimicrobial agents slower. Bacteria positioned in the biofilm in the zone called the non-growing zone are in a good position to survive the antimicrobial agent. This happens due to physiological heterogeneity in biofilms, where the oxygen

and other nutrients are not equally distributed within the biofilm, thus impacting the growth rate of biofilm cells. Most antimicrobial agents depend on growth since the target of these agents is macromolecules. Unexpectedly, they will not have impact microorganisms present in the biofilm that limit macromolecular production. So conventional antibiotics are usually not so effective against slow-growing cells [95; 96; 97].

3. **Persister cells –** Persist or dormant cells are generated stochastically or under endogenous stress, such as exposure to antimicrobial agents. However, these cells are very resistant to being killed by antimicrobial agents. This is possible due to the ability of these cells to slow down or shut down the metabolism, making an ideal situation where these cells are less susceptible to antimicrobial agents due to targets being inactive [93; 94; 97].

4. **Adaptative Responses –** Bacteria have a range of stress responses that enable them to deal with environmental changes, such as oxidative stress, changes in temperature, *DNA* damage, and lower water activity. These responses are present in planktonic cells and in biofilms enhancing bacterial survivability. The difference is that in biofilms, the bacteria have an excellent opportunity to express the characteristics as a result of a slow penetration of the antimicrobial agent or slow growth, allowing biofilms bacteria to respond to the antimicrobial agent, which does not happen in planktonic cells [93; 97].

5. **Phenotypic Heterogeneity –** Biofilms can have cells with different phenotypes and genotypes. This means that the diversity ensures that some individuals will survive in situations of extreme stress, such as the action of an antimicrobial agent. The biofilm cells can switch from one state to other, such as planktonic to persister, to respond to stress, like the environment. Also, they have a heterogeneous population, promoting the variability and division of labor [98; 99].

6. **Quorum sensing –** Quorum Sensing is cell-to-cell communication at the molecular level through chemical signalling molecules called autoinducers. This enables bacteria to recognise the population density by captioning the increase of signalling molecules secreted from community members. The rise in the population density makes the

signal molecules reach adequate levels to activate the maturation and disassembly. In certain situations, such as limited nutrients, it is necessary to provide information through the signals to the bacteria to escape and colonise new niches [97].

2.6 ANTIBIOTIC ADJUVANTS

The role of antibiotics is to kill bacteria or prevent them from reproducing/spreading and not to disrupt or disassemble biofilms. Therefore, antibiotics rarely affect biofilms. Sometimes the only effect they have is on the surface layer of the biofilm, and therefore antibiotics need adjuvants with antibiofilm properties. To eradicate biofilms, antibiotics need adjuvants to improve their action. One of the attributes of antibiotics adjuvants is that they include compounds that can alter antibiotic tolerant cells' physiological states, even in biofilms where they are more resistant [100].

By definition, adjuvants or "helper compounds" are compounds with little or no antibiotic activity that potentiate the activity of antibiotics or block bacteria's primary resistance mechanism when administered or co-administered with antibiotics. Adjuvants can be divided into two classes based on target profile [101]:

- **Class I -** compounds that block antibiotic resistance, through active or intrinsic (passive) inhibition. These adjuvants can be subdivided based on their mechanisms [102]:
  - **Class I.A -** directly inhibit antibiotic resistance mechanism, specifically inactivating enzymes, bypass mechanism.
  - **Class I.B -** bypasses passive or intrinsic resistance mechanisms (such as metabolic pathways or physiological barriers) by increasing antibiotic activity, different from direct inhibition of specific resistance elements, blocking biofilms, facilitating membrane penetration, and others.
- **Class II -** Compounds that do not interact directly with bacteria but interact with the host's defence mechanisms by enhancing antibiotic action [102].

2.6.1 *Antibiofilm Compounds*

As mentioned before, antibiofilm compounds can potentiate the destruction or prevent biofilms. Many approaches with antibiofilm action have already been investigated, such as natural, synthetic, or biological [101].

One of the approaches that have highly broad information is the natural antibiofilm approach. Some examples of these natural compounds are:

- *Populus nigra* and *Populus alba* buds were studied to evaluate their antibiofilm and antimicrobial activity. The extracts of these plants have showed efficient antibiofilm activity against *MRSA* ATCC 43300 and *Staphylococcus aureus* ATCC 6538 with percentages of inhibition grater then 70% for *P.nigra* extracts and more than 50% fot *P.alba* extracts. However, none of them was able to inhibit the biofilm completely. The *P.nigra* also apresented better results in the evaluation of the inhibition of biofilms formation of Bacillus subtilis. The researches think that the biofilm potential can be related to the presence of one or more bioactive coumpounds that apresent antibiofilm properties. So further studies will be needed to isolate and identify these bioactive molecules responsible for the antibiofilms properties [103].
- Kim SW and his partners reported that Ethyl acetate fraction of *Cocculus trilobus* has anti-adhesion effects at the adhesion stage of biofilm formation in Gram-positive bacterias [104].
- Cranberry polyphenols inhibits biofilm formation and prevents the attachment and colonisation of cariogenic and periodontopathogenic bacteria [104].
- *Herba patriniae* extract inhibits biofilm formation and reduced matrix production in *P. aeruginosa* [104].
- Phloretin as an antioxidant is able in apples. Lee et al. observed that reduced biofilm formation in *Escherichia coli* [104].
- Citrus limonoids can interfere with the cell-cell signaling and biofilm formation in *Vibrio harveyi* [104].

To eradicate biofilm-associated infections, antibiofilm compounds must be combined with antibiotics. However, the number of studies testing this combination is not that abundant, and thus there is a lack of knowledge about the possible synergies between these compounds Nevertheless, few studies investigated this kind of dual therapy and the potential of biofilm eradication is quite promising. For instance are:

- aspartic acid and succinic acid were successfully combined with ciprofloxacin and both successufuly inbhit the formation of *P. aeruginosa* biofilms [101].
- *Mentha piperita*, *Cinnammonum zeylanicum*, *Thymus vulgaris*, and *Origanum vulgare* essential oils in combination with antibiotics (norfloxacin, oxacillin, and gentamicin) showed efficacy in the destruction of the biofilm growth of *Staphylococcus aureus* ATCC 29213,*Enterococcus faecalis* ATCC 29212, *Staphylococcus epidermidis* IG4, *Staphylococcus aureus* IG22 [105].
- (+)-Medioresinol isolated from the stem bark of *Sambucus williamsii* in combination with antibiotics (such as ampicillin, cefotaxime, and chloramphenicol) demonstrated that it has antibiofilm activity [106].

3

WORKFLOW IMPLEMENTATION

BIOFILMad *- Catalog of antimicrobial adjuvants to tackle biofilms* was developed as a web application with client-service architecture. It was built as a three-tier application with a presentation tier or client side, a logic tier or server side, and a data tier, corresponding to the database to store articles. The frameworks used to implement the server side were Django and *DRF*, and the client side was React.

Figure 9 shows the workflow structure of ***BIOFILMad*** *- Catalog of antimicrobial adjuvants to tackle biofilms* divided into three four steps, which will be detailed below.

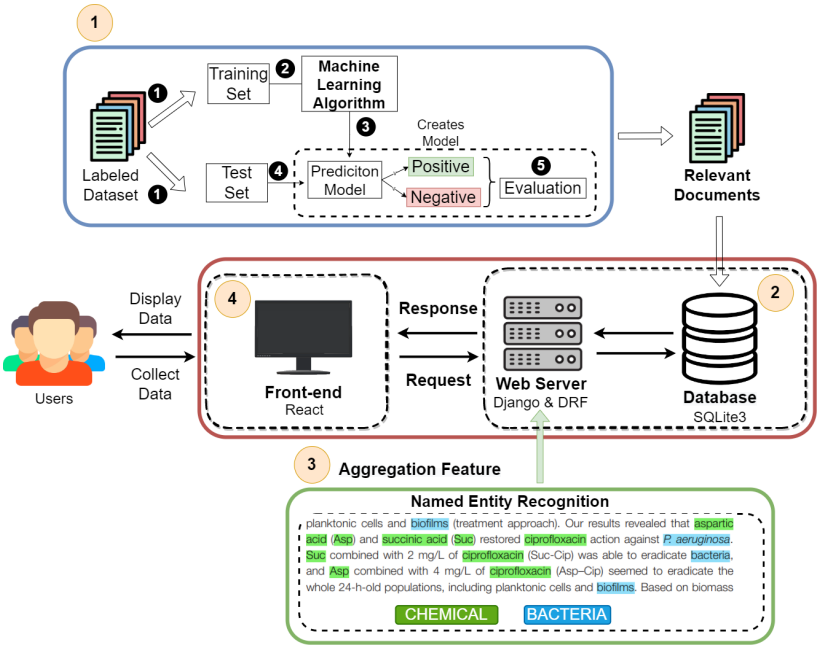


Figure 9: Worflow to develop ***BIOFILMad*** *- Catalog of antimicrobial adjuvants to tackle biofilms.*

As seen in figure 9, the first step in this project is to develop a machine-learning model that allows us to classify relevant articles to the theme. After that implementation, the relevant articles will populate our database and start building the server side with Django and *DRF*. On the server side, there will be a particularity, an aggregation feature where we will use *NER* to classify relevant entities. Finally, we will integrate the client side to demonstrate the relevant information present in the database.

3.1 MACHINE LEARNING

The development of this project was done with Python 3.8, using the conda environment that BioTMPy [107] provided. In Figure 10 it is possible to see the steps and scripts used.

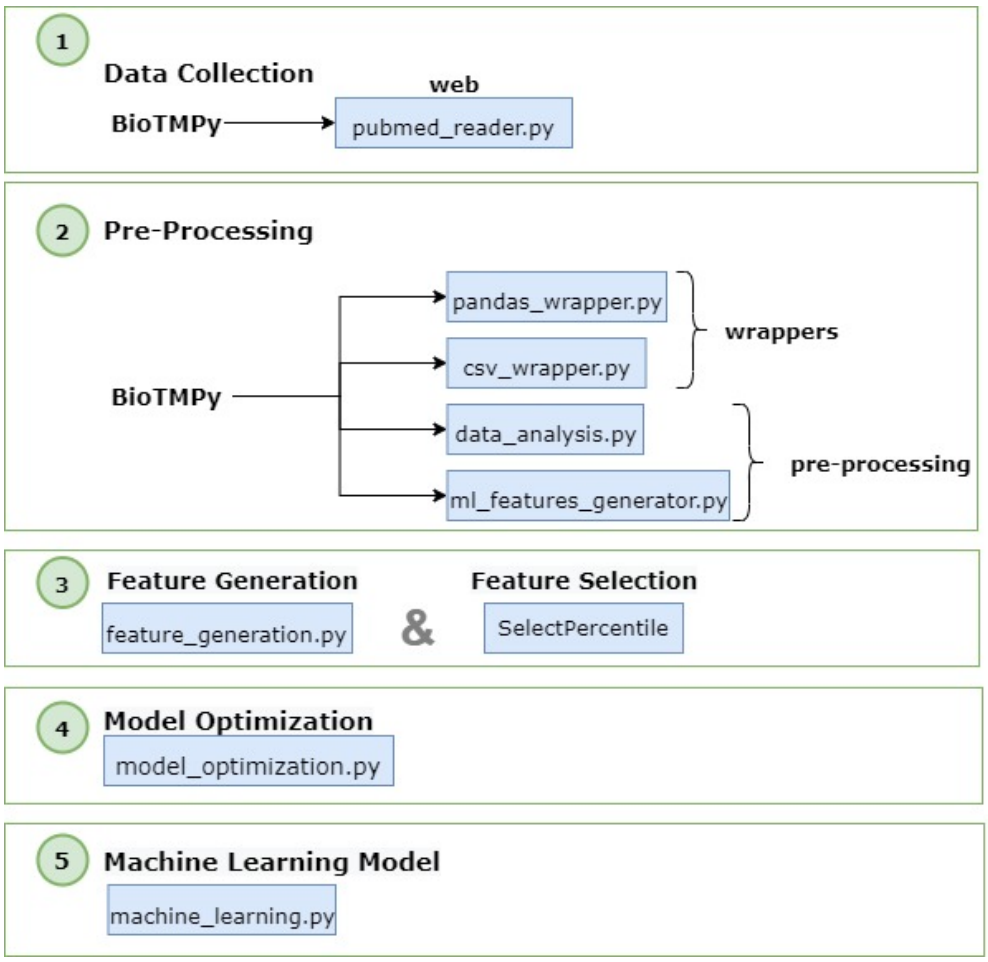


Figure 10: Scheme of the ML model development steps.

3.1.1 *Data Collection*

Bacteria have become increasingly resistant to the available antibiotics, and developing new ones is a process that takes a lot of time. One approach is finding compounds that can restore the activity of existing antibiotics. However, the information about this topic is dispersed in literature, and no database entirely focuses on this topic. Thus, it was necessary to start from scratch and create a dataset to develop the *ML* model. The development of a suitable dataset is an important step when developing a model able to differentiate between relevant and non-relevant articles. For that, the BioTMPy package [107] pubmed_reader.py script was used.

- **pubmed_reader.py** - Script pubmed_reader.py was used to retrieve appropriated data from PubMed a search engine with free access to the MEDLINE database, using BioPython function "*Entrez.esearch*" and "*Entrez.efetch*" [108].

From the pubmed_reader.py three function were used, the "*get_data_from_term*", "*get_data_from_pmid*" and "*pmid_to_docs*".

The first step was to collect a set of keywords that could help obtain articles with relevant information. This set of keywords involved analysing a sample of articles known to be relevant, and the following keywords were selected:

- antibiotic adjuvants biofilms
- resistance breakers biofilms
- antibiotic potentiators biofilms
- anti-resistance drugs biofilms
- biofilm regulators
- biofilm disruptors
- antibiofilm therapeutics

The "*get_data_from_term*" was used to obtain the identifiers of the articles that fit the best with this set of keywords. In total 850 identifiers were retrieved with this function. Following the "*get_data_from_pmid*" function used these identifiers to retrieve the respective title and abstract of PubMed. Finally, the "*pmid_to_docs*" function permitted to transform data into a DOC from BioTMPy Document class.

To transfer the data to a *CSV* another function was used the the "*docs_to_pandasdocs*" present in the pandas_wrapper.py.

3.1.2 *Pre-Processing*

ML algorithms learn from data, so manual curation was required to extract relevant articles, as the algorithm must learn which extracted articles (titles and abstracts) are relevant for training. For being able to perform the manual curation, the content extracted was saved into a *CSV* file. The *CSV* file consists of three columns , title, and abstract, which allows attribution of labels – i.e., relevant or non-relevant – that will later support the training of the supervised models. This was accomplished with "*indexing_series_docs*" and "*docs_to_pandas_docs*" functions from BioTMPy.

Labelling is time-consuming, as all titles and abstracts must be read carefully. Hence, after labelling a few articles manually, labels were assigned programmatically to the rest of the articles. This was accomplished by attributing a score to each document. Articles with lower scores are less likely to be relevant, and those with higher scores are more likely to be relevant. This procedure involved:

- **Convert text to lower case -** python is lowercase sensitive, which means it considers the same word differently if one is uppercase and the other lowercase.
- **Removal of stop words -** Stop words are words that generally do not add much information to a sentence, so they can safely be ignored without sacrificing their meaning and potentially improving performance.

- **Punctuation removal –** Punctuation is removed as it is used frequently in the text (divide sentences, phrases, and paragraphs). These are seen as noise in data as scores are computed according to the occurrence of words, thus punctuation is not relevant.

The first step in assigning these scores was to convert labelled articles into document objects, using the "*csv_to_docs*" function, and getting the relevance (label) of the respective article, using "*dictionary_to_relevances*" function. With the results of these functions, four other functions were used the "*docs_to_pandasdocs*" and "*relevances_to_pandas*" to get the docs and the relevances as pandas to be used in "analysis_dataframe" function. These functions will get the full text, the word count, the number of sentences, and each label and put this information in a data frame. Lastly, the data frame resulting from "*analysis_dataframe*" function will be passed as a parameter of the "*plot_top_n_words*" function generating the top 30 words without stop words.

With the results obtained from the generated plot, three lists were formed, one containing the words from the relevant articles, another from the non-relevant articles, and a list with the word "review" to penalize the articles that are reviewed and discard them since we are not interested in reviews. Subsequently, the respective assigned scores were inserted into a spreadsheet with the articles in descending order of scores, to facilitate the labelling process.

3.1.3 *Feature Generation*

Feature Generation was implemented through the feature_generation.py script to get the maximum amount of information that characterizes the articles by performing, thus adding this information to train the models. Therefore, it will likely result in a more accurate model. The generation of the new features was accomplished through the feature_generation.py.

- **feature_generation.py** - Script was used to create 14 new features that characterize the articles on the subject of this project. To generate the 14 features this script contains 14 functions, each one dedicated to a different aspect.

Table 1 shows the features generated and the respective description in the script feature_generation.py.

Table 1: Name and respective description of the developed features.

Features	Description
Number_of_species	Number of species present in the title of the document
Vowels	Number of vowels present in the document (title + abstract)
Tokens	Number of tokens in the document (title + abstract)
Title_size	Size of the title
Number_stopwords	Number of stopwords in the document (title + abstract)
Upper_letters	Number of upper words in the document (title + abstract)
Lower_letters	Number of lower words in the document (title + abstract)
Nouns	Number of nouns in the document (title + abstract)
Proper_nouns	Number of proper nouns in the document (title + abstract)
Verbs	Number of verbs in the document (title + abstract)
Adjectives	Number of adjectives in the document (title + abstract)
Adverbs	Number of adverbs in the document (title + abstract)
Personal_Pronouns	Number of personal pronouns in the document (title + abstract)
Tokens_mean	Mean of number of tokens present in the document (title + abstract)

In a second approach, to obtain more information to characterize our articles, the calculation of *TF-IDF* was used. This statistical measure evaluates how relevant a word is to an article in a collection of articles by multiplying two metrics: the number of times a word appears in an article and the inverse article frequency across a set of articles; this is particularly useful for scoring words in machine learning algorithms.

3.1.4 *Feature Selection*

This is a crucial step in training a model. Selecting the most relevant features to improve model performance, reduce the complexity of the model, reduce overfitting, and reduce the time spent training. This was accomplished using the "*SelectPercentile*" from Scikit-learn

package [109], selecting the features according to a percentile of highest scores. It takes two arguments the "score_func" and percentile.

Table 2: "*SelectPercentile*" function values tested for each parameter.

	Parameters		
Algorithms	**score_func**	**percentile**	
SVM	f_classif	10	20
Random Forest		10	20
Logistic Regression		10	20

Table 2 shows that for each algorithm the "*f_classif*" function was used for "*score_func*" parameter. This function performs *ANOVA* F-value between feature/label, being suitable for classification tasks. For the "*percentile*" parameter different percentages of features were used in each algorithm to determine their impact on the models' performance, in this case, 10 and 20. Only 10 and 20 were selected because in a previous attempt with 30, 40 and 50 the scores of the metrics in each model were much lower than with 10 and 20.

3.1.5 *Model Optimization*

3.1.5.1 *Models*

One of the most challenging parts of solving a problem is finding a suitable estimator, as different estimators can be better suited for different data types and problems. For this project three different algorithms were used:

- ***SVM* using the "*SVC*" function from Scikit-learn** - can be used for classification (distinguishing between several groups or classes) and applied to both linear and nonlinear problems. *SVM* Classifiers offer good accuracy and perform faster prediction and use less memory because they use a subset of training points in the decision phase.

- **Logistics Regression using the "*LogisticRegression*" function from Scikit-learn** - is one of the simplest and most commonly used Machine Learning algorithms for binary classification problems. The logistic regression presents a sigmoid function, called the logistic function, which can take any real number and map it between 0 and 1, giving an S-shaped curve. If the output of the sigmoid function is higher than 0.5, we can classify the outcome as 1, and if it is less than 0.5, we can classify it as 0, so if our output is is 0.75, we can say in terms of the probability that there is 75% chance that the article will be relevant.
- **Random Forest using the "*RandomForestClassifier*" function from Scikit-learn** has a variety of applications, and one of them is classification. Decision trees are generated on a randomly split dataset and this collection of decision tree classifiers is also known as the forest. In a classification problem, each tree votes and the most popular class is chosen as the final result.

3.1.5.2 *Performance Evaluation*

The "*StratifiedKFold*" function from the Scikit-learn package [109] was used to perform the stratified 5-fold cross-validation, as this function preserves the percentage of samples for each class in train and test sets. A CV with 5-folds was applied by the function "*cross_validate*", allowing to specify multiple evaluation metrics. The metrics chosen to determine the performance of the models were the 'f1 score', 'accuracy score', 'recall score', and 'precision score' from the Scikit-learn package [109]:

- **'f1 score'** can be interpreted as a weighted average of the precision and recall; , **'accuracy score'** is the exact match of the labels predicted that correspond to the actual labels.
- **'recall score'** is the ratio TP / (TP + FN), where the recall is the capability of the classifier to find all the positive elements.
- **'precision score'** is the ratio TP / (TP + FP), where the Precision is the capability of the classifier not to label wrongly as positive an element that is negative.

In addition, confusion matrices were created using the function "*plot_confusion_matrix*", after applying the "cross_val_predict" function with 5 folds.

3.1.5.3 *Optimization*

Model hyperparameters must be improved in order to increase the model's performance. The best models hyperparameters were selected using the model_optimization.py were it is applied the Grid method with CV to the selected models, using "*GridSearchCV*" function from the Scikit-learn package [109]. This function performs an exhaustive search over the specified parameters' values for an estimator, thus determining the best parameters for each model. These test values presented in table 3 were selected experimentally.

Table 3: Hyperparameters of each model and values tested for each parameter.

Algorithm	Parameters	Values Tested
SVM	'C'	1, 10, 20, 100, 1000
	'gama'	auto, 0.001, 0.012, 0.1
	'kernel'	linear, rbf
Random Forest	'max_depth'	3,5,10
	'min_samples_split'	2,5,10
	'min_sample_leaf'	2,5,10
	'n_estimators'	20, 50, 100
Logistic Regression	'solvers'	'newton-cg', 'lbfgs', lib-linear'
	'penalty'	'none', 'l1', 'l2'
	'c_values'	100, 10, 1.0, 0.1, 0.01

Another parameter included the random state uses the function "*randrange*" from the random package [110] to generate a number between 0 and 2147483647, allowing to determine which seed was used to obtain the accuracy, precision, recall and f1-score. Therefore the results can be replicated by using the same seed.

3.2 BACKEND

The ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* backend application is a web application programming interface (API), developed using Django *DRF* that allows relevant data and functionalities to be available via the internet and that can be accessed, for instance, by a Web frontend application that displays the data to a user in a web browser. It is built upon Django and *DRF*, with several other components like: DRF authoken to provide token-based authentication, an aggregation feature using spacy and SciSpaCy to aggregate relevant data, a database layer that uses SQLite3; and a biopython Entrez module that is used to obtain external data based on user input.

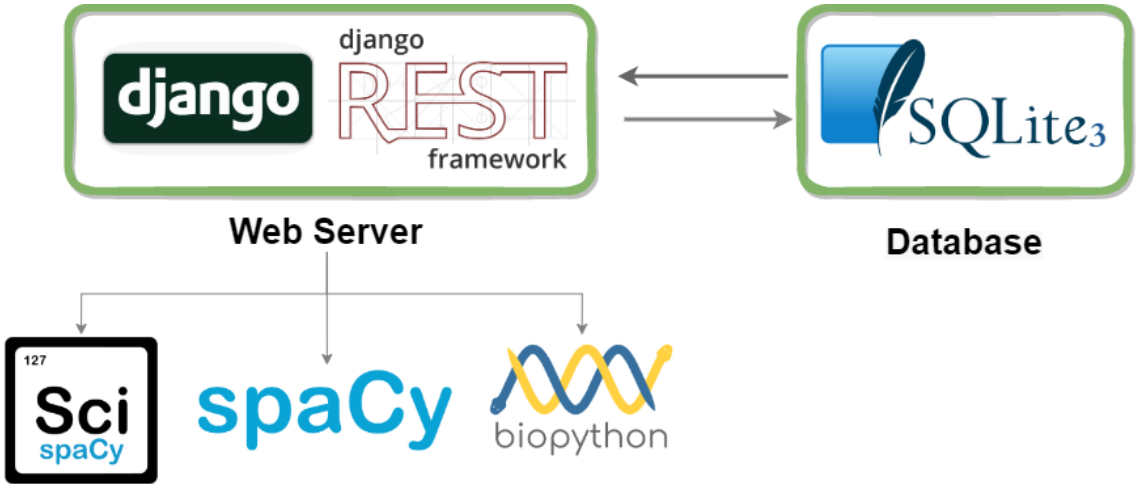


Figure 11: Application backend implementation.

This figure shows the respective frameworks to implement the back-end of this application. Also presents the main packages used to implement the aggregation feature.

3.2.1 *Database Models*

The ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application uses an SQLite3 database system in order to store and interact with the data. SQLite3 meets the desired requirements for the application functionality but it may be necessary for the future to use a more scalable database technology. Django automatically gives a database-

abstraction API that allows the creation, retrieval, update and deletion of objects for multiple database systems, which eases the task of changing to a more reliable and scalable database in the future. In particular, all models created as abstractions of the database tables will remain the same independent of the database used.

For the construction of this application 18 models were created to incorporate important information extracted from PubMed through the biopython Entrez module, and information generated by the aggregation module. Figure 12 shows an example of the Author model. An Entity Relationship diagram is shown in Appendices, section A.2, illustrating the established database tables.

```
class Author(models.Model):
    identifier = models.CharField(max_length=30)
    forename = models.CharField(max_length=30)
    lastname = models.CharField(max_length=30)
    initials = models.CharField(max_length=10)
    collectivename = models.CharField(max_length=30)
    affiliationInfo = models.ManyToManyField(AffiliationInfo)
```

Figure 12: Author model present in ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application.

Figure presents the Author model with six fields: identifier, forename, lastname, initials, collectivename and affiliationInfo.

3.2.2 *Django REST Framework*

In order to make data, aggregations and the ML framework itself accessible to several different platforms, an *API* was built using *DRF* that makes it easy to provide all the relevant data and functionalities via a *JSON API*. To construct the *API* with *DRF* there were three main steps taken:

- **Create the serializers** to convert complex data (model instances, querysets,etc) into Python datatypes.
- **Create the viewsets** to return data to the user in *JSON* format.

- **Creating the endpoints of *API*** to make the viewsets available.

In specific, for rendering JSON based models the *ModelViewSet* was used and for custom content the *APIView*, e.g the download and upload of *CSV* files to be used by model framework. Figure 29 in the Appendices, section A.3 demonstrates the *DRF* view inheritance diagram. *DRF* provides a powerful visualization of the data in *JSON* format at the *API* endpoint helping visualise the data. The *JSON* returned by this request will be rendered on the web page, as seen in figure 13.


```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "count": 228,
    "next": null,
    "previous": null,
    "results": [
        {
            "id": 1,
            "article_type": "Article",
            "pmid": "29661867",
            "eLocationID_article": "10.1128/AAC.00146-18",
            "title": "Triclosan Is an Aminoglycoside Adjuvant for Eradication of Pseudomonas aeruginosa Biofilms.",
            "abstract": "One of the most important clinical obstacles in cystic fibrosis (CF) treatment is antibiotic
            "datecompleted": "2019/08/19",
```


Figure 13: API Articles View.

Figure shows information of "*Triclosan Is an Aminoglycoside Adjuvant for Eradication of Pseudomonas aeruginosa Biofilms*" [111] in *JSON* format.

3.2.2.1 *Django REST Framework authtoken*

The Upload and Download of data should only be done by authenticated users. *DRF* authtoken adds *DRF* the capability of token based authentication. Token-based authentication is a good authentication method because allows for a generic authentication across platforms (e.g. browser, mobile apps, etc.). For example, a cookie-based authentication only makes sense for browser-based applications. For the routes that need permission to be accessed @authentication_classes and @permission_classes decorators are used. *DRF* and *DRF* authtoken provide decorators that are used to authorize access to private routes. These decorators guarantee that only authenticated users can use the *API* routes.

3.2.3 *Entrez module*

Authenticated users can upload data to the application in a *CSV* format with *PMID*s to be evaluated - by the *ML* framework - and added to the database. To get additional information from the *PMID*s supplied by the user, the Entrez module is used to communicate with PubMed *API*. An example of the retrieved data is presented in figure 14.

```
{
    "PubmedArticle": [{
        "MedlineCitation": {
          "SpaceFlightMission": [],
          "KeywordList": [
            ["Pseudomonas aeruginosa",
            "biofilm",
            "persister",
            "tobramycin",
            "triclosan"
            ]],
          "PMID": "29661867",
    ...]
    "PubmedBookArticle": []
}
```

Figure 14: Information extracted from PubMed through the Entrez module [108].

Figure shows pseudo information extracted from PubMed about the article "*Triclosan Is an Aminoglycoside Adjuvant for Eradication of Pseudomonas aeruginosa Biofilms*".

The resulting data from the Entrez module is mapped to the correct Django models, such as Article, Author, Keyword, etc and inserted into the database.

3.2.4 *SciSpacy and Aggregation Feature*

spaCy [112] is an open-source library that implements the algorithms to process and analyze data in *NLP* tasks with Python. It uses container objects that characterize elements

of natural language texts (sentences, words, labels, etc.), and has established steps to extract information from a text. These consist of certain chained components, acting on the text forming a DOC object that contains the extracted structured data from the text [113; 114].

The language models, which are among the most remarkable features of spaCy [112], allow performing *NLP* tasks such as *NER*. *NER* identifies entities present in the text. For this project theme, the SciSpaCy library was used, as satisfies the text processing, focusing on the biomedical field, and containing spacy models. SciSpaCy is composed of two main packages [115]:

- **en_ core_ sci_ sm -** Full pipeline for biomedical data.
- **en_ core_ sci_ md -** Full pipeline for biomedical data with a larger vocabulary and 50k word vectors.

A table with the total vocab size, vector count, min word freq and min doc freq is presented in Appendices, section A.4.

Since the goal is to give relevant information about antibiotics/antimicrobial, bacteria/disease and compounds the en_ ner_bc5cdr_md model seems to meet all the requirements, since identifies **DISEASE** and **CHEMICAL** entities. A table with all the models of SciSpaCy is presented in Appendices, section A.5.

Figure 15 shows an example of the identification of the entities.

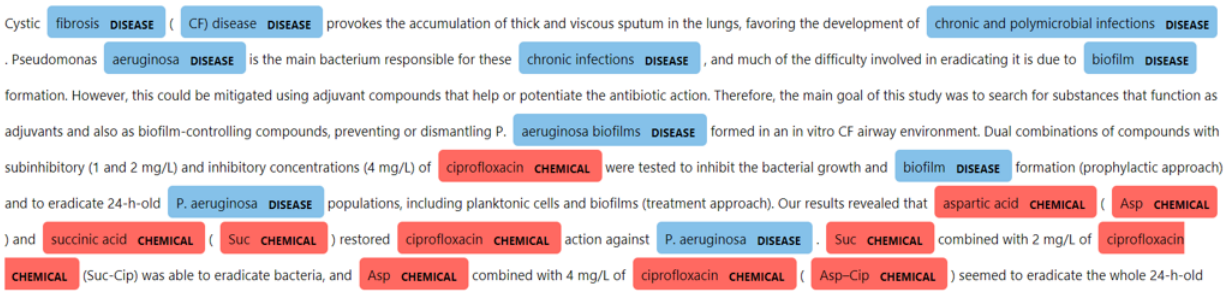


Figure 15: Identification of the entities thrown by the en_ner_bc5cdr_md model.

This text was taken from the publication "*Fostering Innovation in the Treatment of Chronic Polymicrobial Cystic Fibrosis-Associated Infections Exploring Aspartic Acid and Succinic Acid as Ciprofloxacin Adjuvants*" [101]. The words encircled with red are chemicals, and with blue is a disease.

The CHEMICAL entity was divided into *Antimicrobial Compounds* and *Antibiotics/Antimicrobials* since the SciSpaCy classifies both of them as chemicals. These entities extracted with SciSpacy were used to construct the Aggregation feature. This was accomplished with the creation of two models, where in one each entity is saved and in the second the relation between two entities at the article level is made. This allows the users to know for specific entity found by SciSpaCy which other entities are related to that one and in which article that happens. To improve the ScispaCy entity detection, the python library Inflect was used to join singular / plural words in one entity (e.g SciSpaCy may detect *Pseudomona* and *Pseudomonas* as two separte entities).

3.3 FRONTEND

The frontend of the ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application is a web based user interface that gets data from the Django backend *API*. It is a standalone web application that can be deployed to any web server and was built using React and Material-UI component library that provides pre-built *UI* components like Buttons, Icons and DataGrids (figure 16. It is important to highlight three main aspects of ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms*'s frontend, namely its structure, the client-side routing and its authentication capabilities.

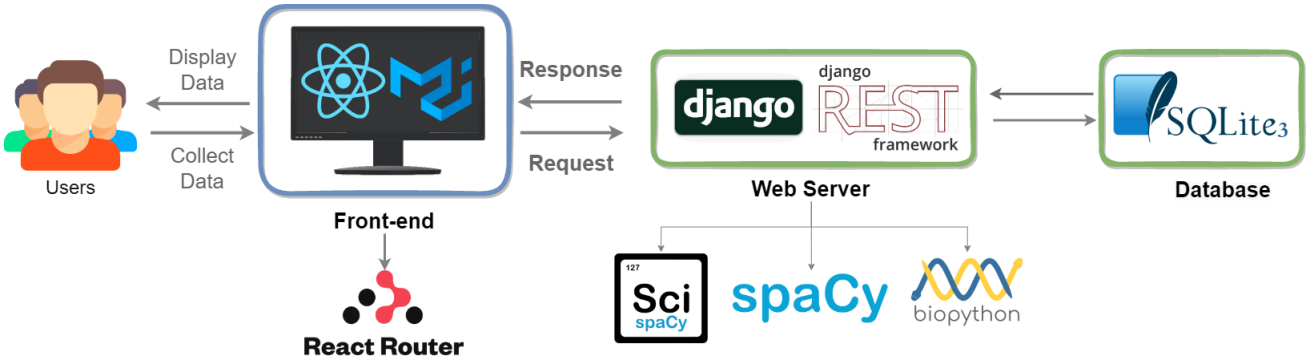


Figure 16: Application architecture with front-end and back-end implementation.

This figure shows each tier of the application with the respective frameworks, database used and the main packages implemented.

3.3.1 *Structure*

To provide the user with the relevant information and functionalities, the application is composed of nine main routes/pages, mapped to nine react components:

- **HomePage** is a page that contains a description of application and a search bar.
- **Articles** displays and allows search's and filtering of the articles present in the database.
- **Article** shows the content of specific article.
- **About** contains information about the purpose of the application and the team members' contacts.
- **Antimicrobial Search** List all antimicrobial entities- compounds, diseases - extract from the articles by the ScispaCy component of the backend.
- **Antimicrobial Statistics** shows relations between entities.
- **Login** to authenticate a user.
- **LogOut** to logout a user.
- **Model** allows authenticated users using to interact with the ML framework.

3.3.2 *Routing*

The frontend is *SPA*, a route-based navigation tool, meaning that part of the interactions of the user are passed as parameters in the URL, which allow users to access the functionality of the application directly. For example, accessing an *URL* like www.adjuvants-snap-url.com/entities-stats/pseudomonas will always show the latest entity relations for pseudomonas and the user can save that URL as a favourite in the browser and use it later. To ensure a seamless user experience during navigation a browser-only routing react-router-dom is used, and for getting data from the *API* asynchronously the promised

based *HTTP* client Axios [116] is used. Figure 17 shows multiple routes to determine which component should render based on the clicked link.

```
import React from "react";
import { BrowserRouter as Router, Routes, Route } from "react-router-dom";
function App() {
  return (
        <Router>
              <Routes>
                <Route path="/" exact element={<Home />} />
                <Route path="/about" element={<About />} />
                <Route path="/article/:id" element={<Article />} />
              </Routes>
        </Router>
  );
}
```

Figure 17: Example of multiple routes of the ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* frontend using the react-router-dom.

3.3.3 *Authentication*

The interactions with the ML framework should only be available to authenticated users. The *Login* React component was created to provide an authentication form where the user can provide the credentials used to authenticate against the login *URL* of the backend. The backend responds with the authtoken saved on the local storage. The authtoken is added to the *HTTP* requests header made to the backend *API* when required (e.g when uploading a new model for the *ML* Framework). The *LogOut* component is an interfaceless component that erases the token from the local storage and redirects the user to the HomePage.

4

RESULTS AND DISCUSSION

This chapter presents the results and evaluation of the *ML* algorithms and the final result of the ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application. First, a description of the results of ML is provided as well as an interpretation. Then it is described each page of the application and its respective functionality and aspect.

4.1 MACHINE LEARNING

Creating a model to classify relevant articles involves determining which ML algorithms are suitable for the data type. Three different models were created and evaluated. Manual curation was implemented as part of the data pre-processing to create a dataset and accomplish the goal of training a model to classify relevant articles. Thus, the features influence the model's performance, so feature generation and feature selection were applied. The models were further optimized, and the best parameters were selected and applied to the training process.

4.1.1 *Data Pre-processing*

After getting the data from PubMed to create a dataset manual curation was done. However, labeling is a time consuming task, so labels were assigned programmatically with

the help of the "*plot_top_n_words*" function from BioTMPy package resulting in the following graphics:

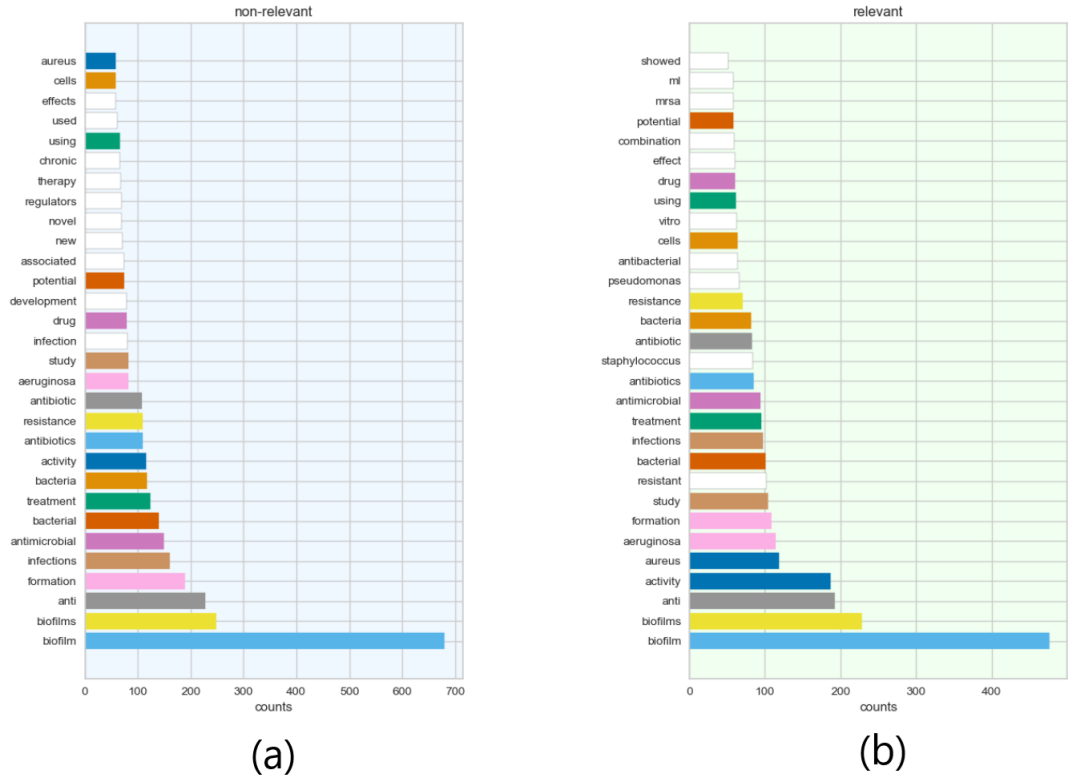


Figure 18: Top 30 Words (without stopwords) created with BioTMPy package.

This figure shows a graphic withe the top 30 words present in the (a) relevant articles or (b) non-relevant articles.

These graphs indicate the top 30 words present in relevant and non-relevant articles. The focus is on the words with a white bar, as this color shows that these words are only present either in the relevant articles (Figure 18 a) or in non-relevant articles (Figure 18 b). With these results, three lists were formed to help attribute the respective label to each document faster, saving some time in the labeling process to train the models further.

4.1.2 *Feature Generation & Feature Selection*

The goal of performing feature generation was to get the maximum amount of information that characterizes our articles, thus adding this information to train the models. Therefore, it will likely result in a more accurate model. The created futures are presented in Table 1, section 3.1.3. These features were made to give value or emphasize certain aspects that our dataset presents. Of the features created, the one that has the most impact and adds the most value is the Number_of_species since it goes to each title and abstract and counts how many species are mentioned. The remaining features can be said to be more general. A good approach in the future would be to add more features, such as the number of antibiotics or antimicrobials or the number of compounds present in the respective title and abstract. Also, in a second approach, to obtain more information to characterize the articles of the dataset, the calculation *TF-IDF* was used.

Feature Selection was implemented to improve the scores for each metric since the scores were always low without it. Table 4 shows the scores for each metric without feature selection.

Table 4: Scores of the selected models obtained with the metrics of the sklearn package without feature selection.

	Without Features			
	Accuracy	**Precision**	**Recall**	**F1 Score**
SVM	0.60	0.54	0.30	0.39
Random Forest	0.58	0.61	0. 27	0.37
Logistic Regression	0.68	0.63	0.30	0.42

The results demonstrate that probably there were a lot of redundant features to be considered to train our models. With the implementation of the feature selection, the power of prediction of the algorithms was increased, and the process became more accurate, as seen in Table 7.

Regarding the features selected by the "*SelectPercentile*" from Scikit-learn package [109], when the parameter "*percentile*" is 10% of the features selected we have a total of 945 features and in 20% we have 1890 features selected.

Table 5: List of the first 50 features selected by "*SelectPercentile*" function from Scikit-learn package and the number of total features.

Percentile	First 50 Features Selected	Total of Features Selected
10 %	'gram', 'tokens_title', 'tokens_total', 'stats_mean', 'stats_min', 'sentence_size', 'upper_letters', 'nouns', 'token_stats_mean', "cv026", , "staphylococcus", "mrsa", "fungicida", "edta", "glycopeptide", "resistant", "disease", "virulence", "approaches", "pseudomonas", "staphylococci", "eradicated", "klebsiella", "albicans", "patients", "biofilms", "adjuvant", "adaptation", "aeruginosa", "inhibiting", "potentiator","disrupters", "266", "mutations", "tobramycin", "polyethylenimine", "pneumoniae", "aureus", "inhibition", "7a", "bacteria","cystic", "therapy", "antibacterial", "inhibits", "flavonoids", "antibiofilm", "disrupting", "combined", "potentiators", "regulator"	945
20 %	'gram', 'tokens_title', 'tokens_total', 'stats_mean', 'stats_min', 'sentence_size', 'upper_letters', 'nouns', 'token_stats_mean', "transcription", "biofilm", "methylthiazolyldiphenyl" "ncib3610", "patients"", "disruption", "je2", "potentiator'", "098'", "ciprofloxacin", "staphylococcus", "bacterial'", "726'",, "irradiated'", "biofilms", "43300", "rifampin'", "miconazoctylium'", "biofilm", "pao60'", "fibrosis", "352083", "regulator", "aeruginosa'", "02", "7", "pseudomonas", "adjuvant", "rna" , "91'", "research'", "inhibit", "cryptococcus", "1620", "edta", "vitro", "col", "fd2", "potentiate", restores", "ppy"	1890

This table presents the first 50 features selected to train our models since the number of total features is too large to be presented in this table.

From what it is possible to observe from the results with the small sample of features selected presented in Table 5 taking into account the total number of features from each percentile, it is possible to notice that it can add value to the training of the models. From the 50 first features the "staphylococcus", "aureus", "pseudomonas", "aeruginosa", "adjuvant", "biofilms", "regulator", "inhibition", "bacteria", "antibiofilm", "antibacterial", "resistant", "potentiators", "approaches" are some of the selected features.Bearing in mind the theme of this project, it is possible to make a connection between these features and point to the problem at hand. For example, Staphylococcus aureus and Pseudomonas aeruginosa were classified as antibiotic-resistant "priority pathogens" by the WHO since they become

resistant to a large number of antibiotics. Also, the goal is to find potentiators/antibiofilm compounds to combat the resistance of bacteria to antibiotics/antimicrobials. As expected, only some selected features will be meaningful for training the models. Table 5 presents "je2", "1620", and "fd2" features that are not proper to the theme and would not help much to train the model.

4.1.3 *Model Optimization*

The hyperparameters of the selected models were optimized to obtain the best models for predicting the relevant articles, using the the methods of GridSearchCV. The hyperparameters optimized for each model and the values tested for each parameter are described in Table 8 (section 3.1.X). The table 6 presents the parameters values that originate the best results for each model using GridSearchCV. Also, the means test score, precision, and recall, accuracy are presented for the optimized models.

Table 6: Values selected for each hyperparameter and mean of scores, Precision and Recall for each optimized model GridSearchCV from Sckit-Learn.

			10 % of Features			**20 % of Features**		
Algorithm	**Parameters**	**Selected Values**	**Mean test score**	**Precision**	**Recall**	**Mean test score**	**Precisison**	**Recall**
SVM	**'C'**	100	0.82	0.83	0.74	0.75	0.80	0.68
	'kernel'	'linear'						
RF	**'max_depth'**	10	0.84	0.70	0.52	0.78	0.77	0.60
	'n_estimators'	20						
LR	**'C'**	100	0.85	0.84	0.70	0.68	0.70	0.61
	'penalty'	'l2'						
	'solver'	'liblinear'						

In Table 6, shows the algorithms and the respective parameters selected for the different percentiles of features by the "*GridSearchCV*" function. It's possible to denotate that in general the hyperparameters obtained better results with 10% of the selected features.

4.1.4 *Models Performance*

The dataset formed througth the data collection were used to create models to classify if a article is relevant or non-relevant. The dataset formed througth the data collection were used to create models to classify if a article is relevant or non-relevant. The algorithms *SVM*, *RF*, and *LR* were used in to train the model with a different percentile of features selected and the evaluated by a 5-fold CV. The models with "– 1" have 10% of all features, while those with "– 2" have 20% of all features. Table 3, shows the models scores to each metric.

Table 7: Scores of the selected models obtained with the metrics of the sklearn package.

Model	Algorithm	Accuracy	Precision	Recall	F1 Score	Features
SVM – 1	Support Vector Machine	0.7514	0.6595	0.5438	0.5961	945
RF – 1	Random Forest	0.7633	0.6666	0.3333	0.4444	945
LR – 1	Logistic Regression	**0.8461**	0.7837	**0.6170**	**0.6904**	945
SVM – 2	Support Vector Machine	0.7810	0.7777	0.2978	0.4307	1890
RF – 2	Random Forest	0.7988	**0.793**	0.4509	0.575	1890
LR - 2	Logistic Regression	0.7159	0.6666	0.2545	0.3684	1890

The analysis of results shows that the models with 20% of features selected generally have better scores. However, the *LR* - 1 model stands out for having higher accuracy, recall, and precision. Only the precision is slightly below the *RF* - 2 model.

Confusion matrices were created to mainly determine the number of fn provided by each model, as shown in tables 8 and 9.

Table 8: Confusion Matrix of the optimized *SVM* - 1, RF -1 and LR - 1 models, with prediction values for 10% of features selected.

	10% of Features Selected					
	SVM - 1		**RF - 1**		**LR - 1**	
Predicted / Real	**Relevant**	**Non-Relevant**	**Relevant**	**Non-Relevant**	**Relevant**	**Non-Relevant**
Relevant	31	26	16	32	29	18
Non-Relevant	16	96	8	113	8	114

The model attributes the classification through 0 and 1. To show the results that were substituted with Relevant and Non-Relevant, respectively.

Table 9: Confusion Matrix of the optimized SVM - 2, RF -2 and LR - 2 models, with prediction values for 20% of features selected.

	20% of Features Selected					
	SVM - 2		**RF - 2**		**LR - 2**	
Predicted / Real	**Relevant**	**Non-Relevant**	**Relevant**	**Non-Relevant**	**Relevant**	**Non-Relevant**
Relevant	14	33	23	28	14	41
Non-Relevant	4	118	6	112	7	107

The model attributes the classification through 0 and 1. To show the results that were substituted with Relevant and Non-Relevant, respectively.

The results presented in tables 8 and 9, shows that the model with more *FN* was *LR* - 2. Taking in consideration that the objective of the models is to correctly classify articles about antibiotics adjuvants, having a higher number of *FN* is not the best option, since it essential to identify the articles that can give important information to the researches.

Considering all the models, the *LR* - 1 is the model with most *TN* and the second with most *TP*, meaning that it was the best to predict the correct labels. Also, it is the model with fewer *FN*, which is a good pint because this model does not misclassify the real relevant articles present in the dataset, even if that costs some of the *FP*. These results confirm that, the *LR* - 1 model is the most suitable model to perform the classification task.

4.2 BIOFILMAD - CATALOG OF ANTIMICROBIAL ADJUVANTS TO TACKLE BIOFILMS INTERFACE

The ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application's primary purpose is to let users navigate to see interesting content in searching for antibiotics adjuvants with antibiofilm activity, being accomplished with the server side and the client side.

When entering the ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application, the user will first have contact with the Homepage (figure 19 that introduces a Navigation bar, a description, and a road map through the application. The goal of the Homepage is to let the user know more about the problem that we face nowadays and to let

the user know how he can use this application. There is also present a search bar on top of the Homepage. This search bar can redirect the user to two places:

- Suppose the user's input in the search bar is an entity available in our database table. In that case, the user will be redirected to a page where the aggregation of that entity is presented.
- If the input is unavailable, it will redirect the user to the articles page, which will show the articles where the input appears. Since the relation between entities is made with *NLP*, it is not certain that it will identify every intended entity in the text. This is the motive for the user will be redirected to the articles in case of identification failure.

This search bar has the particularity of having an auto-complete function, so if an entity is presented in the database, the user will know immediately.

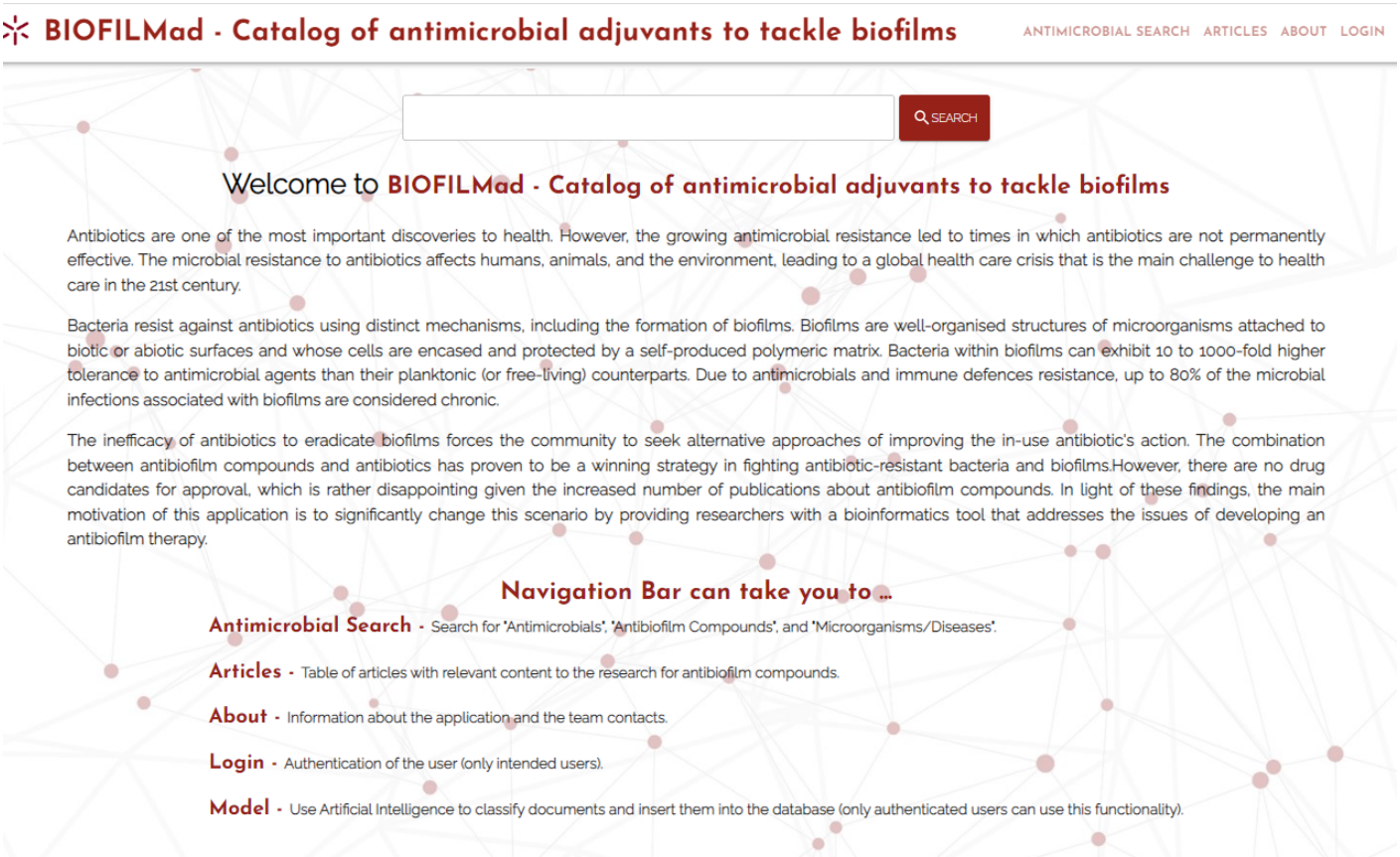


Figure 19: ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application Homepage.

Heading now to the tabs of the application (figure 20), we can see four tabs.

Figure 20: Navigation bar of ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application with four tabs.

Let us start with the About tab. The About tab redirects the user to a page where it is possible to find an insight into what motivated the development of this application, letting the user know the issue and how this tool can help them. It also presents the contacts of the team involved in this development, as seen in figure 21.

Figure 21: ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application About page.

The Antimicrobial Search tab redirects the user to a page that presents three sections: "Microorganism/Disease", "Antibiofilm Compounds," and "Antimicrobials". Each of these sections has a search bar and respective examples of how to search entities. The goal of this page is to let the user search directly by a specific type of entity. Suppose the user searches for "*pseudomonas*" on the "Microorganism/Disease" search bar. Three results appear, as seen in figure 22. By clicking on one of these results, the user will be redirected to a page where it is possible to see the aggregation of that entity.

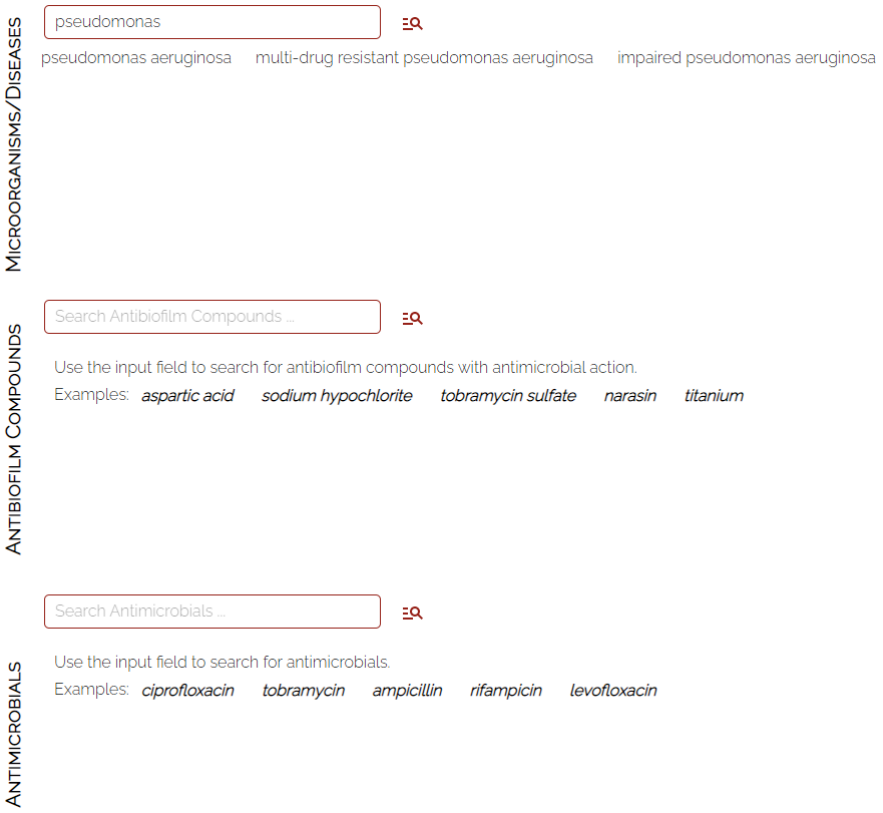


Figure 22: ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application Antimicrobial Search page.

Figure 23 shows an example of an output of a search by "*pseudomonas aeruginosa*". The page to the user comprehends the entity's name, the number of articles where the entity appears, and the relations between the other entities listed in alphabetic order. This way, the user can have an idea of which compounds and "Antimicrobials" the "*pseudomonas aeruginosa*" entity is related to. Below each entity is the title of the articles where the entity emerges,

and clicking on it will redirect the user to the specific article to see more information about that study. It is also possible to click on another entity to see its statistics.

Figure 23: ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application aggregation page.

Figure shows a pseudo relati of the "*pseudomonas aeruginosa*" with antimicrobials.

Proceeding to the Articles tab delivers a table with the articles saved in our database (figure 24). There are two ways to proceed to this page: via the search bar on the Home page or by clicking on the tab Articles. This page also has a search bar if the user wants to search directly for an article with its title, phrase, or merely by a specific word(s), getting a result that will fit the content searched. Also, the left side of the page has a filter that enables the user to filter the articles by type if only a specific article is required, making the search easier by filtering it.

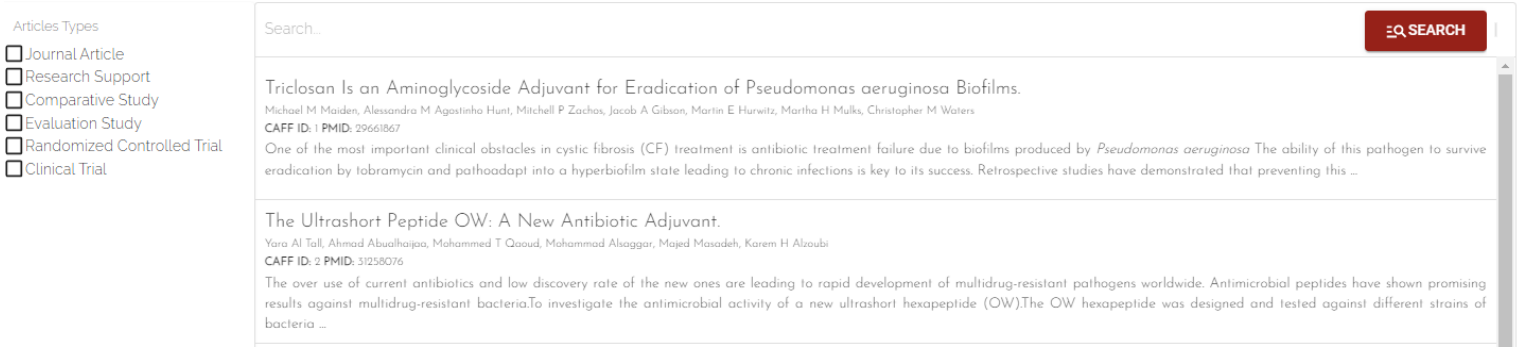
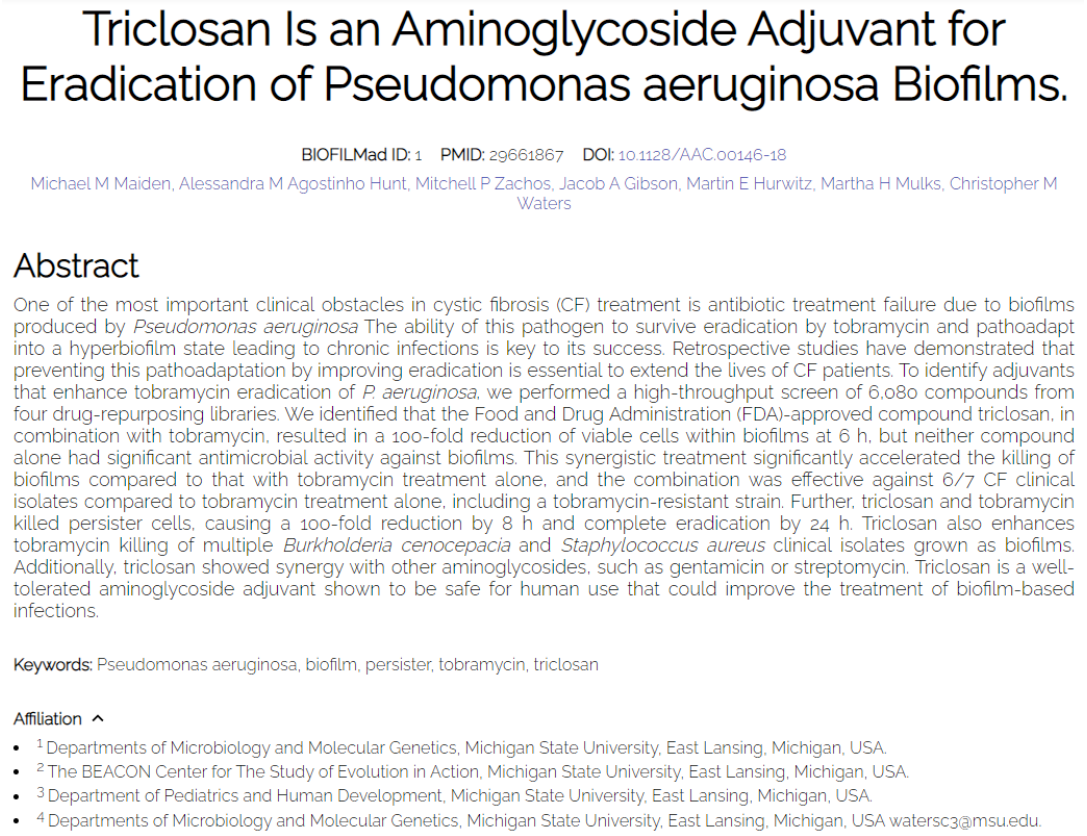


Figure 24: ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application articles page.

Articles page demonstrate the articles present in the database with a search bar.

A click on an article from the table will redirect the user to the particular article (figure 25), presenting the title, authors, keywords, abstract, etc. This page has some particularities, such as:

- Clicking in a will redirect the user to the *URL* or services related to a single resource.
- Clicking on the author will redirect the user to the articles page only with the specific author's articles.
- Clicking on *MeSH* terms will redirect the user to the National Library of Medicine and return the results for that term.



Triclosan Is an Aminoglycoside Adjuvant for Eradication of Pseudomonas aeruginosa Biofilms.

BIOFILMad ID: 1 PMID: 29661867 DOI: 10.1128/AAC.00146-18

Michael M Maiden, Alessandra M Agostinho Hunt, Mitchell P Zachos, Jacob A Gibson, Martin E Hurwitz, Martha H Mulks, Christopher M Waters

Abstract

One of the most important clinical obstacles in cystic fibrosis (CF) treatment is antibiotic treatment failure due to biofilms produced by *Pseudomonas aeruginosa* The ability of this pathogen to survive eradication by tobramycin and pathoadapt into a hyperbiofilm state leading to chronic infections is key to its success. Retrospective studies have demonstrated that preventing this pathoadaptation by improving eradication is essential to extend the lives of CF patients. To identify adjuvants that enhance tobramycin eradication of *P. aeruginosa*, we performed a high-throughput screen of 6,080 compounds from four drug-repurposing libraries. We identified that the Food and Drug Administration (FDA)-approved compound triclosan, in combination with tobramycin, resulted in a 100-fold reduction of viable cells within biofilms at 6 h, but neither compound alone had significant antimicrobial activity against biofilms. This synergistic treatment significantly accelerated the killing of biofilms compared to that with tobramycin treatment alone, and the combination was effective against 6/7 CF clinical isolates compared to tobramycin treatment alone, including a tobramycin-resistant strain. Further, triclosan and tobramycin killed persister cells, causing a 100-fold reduction by 8 h and complete eradication by 24 h. Triclosan also enhances tobramycin killing of multiple *Burkholderia cenocepacia* and *Staphylococcus aureus* clinical isolates grown as biofilms. Additionally, triclosan showed synergy with other aminoglycosides, such as gentamicin or streptomycin. Triclosan is a well-tolerated aminoglycoside adjuvant shown to be safe for human use that could improve the treatment of biofilm-based infections.

Keywords: Pseudomonas aeruginosa, biofilm, persister, tobramycin, triclosan

Affiliation

- [1] Departments of Microbiology and Molecular Genetics, Michigan State University, East Lansing, Michigan, USA.
- [2] The BEACON Center for The Study of Evolution in Action, Michigan State University, East Lansing, Michigan, USA.
- [3] Department of Pediatrics and Human Development, Michigan State University, East Lansing, Michigan, USA.
- [4] Departments of Microbiology and Molecular Genetics, Michigan State University, East Lansing, Michigan, USA watersc3@msu.edu.

Figure 25: ***BIOFILMad*** *- Catalog of antimicrobial adjuvants to tackle biofilms* application article page.

Figure shows pseudo information about "*Triclosan Is an Aminoglycoside Adjuvant for Eradication of Pseudomonas aeruginosa Biofilms*" [111].

Figure 26 shows the content presented by clicking on the Login tab. The Login was implemented for specific users to utilize the model developed to classify articles as relevant or non-relevant. Because it was decided that only a particular user can use this functionality,

there is no option to create a new user. Only the administrator of the ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application can do it.

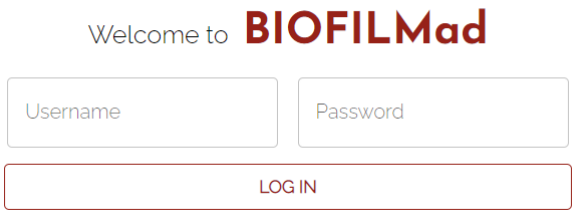


Figure 26: ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application log in.

When the user logs in successfully, the user is sent to the Home Page. The navigation bar now shows the Model tab, which redirects the user to a page where it is possible to see a description of how to use the model (figure 27). This page presents three buttons:

- **Upload *CSV* -** where the user can upload a *CSV* with a specific format to the model classifies.
- **Download *CSV* -** where the server side returns a *CSV* with an additional column with the respective classification.
- **Update Data -** where the user can insert information on the database.

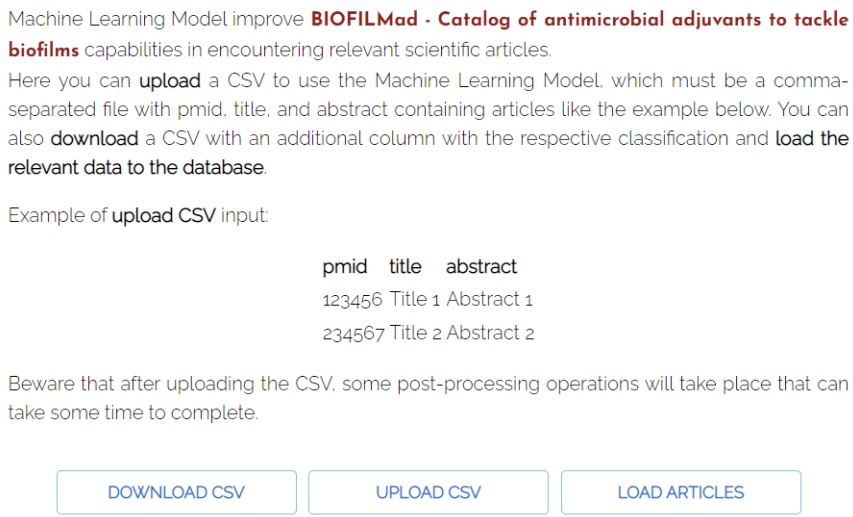


Figure 27: ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* application model page.

Model page that permits the user to classify articles and to load data to the database.

5

CONCLUSIONS, LIMITATIONS AND FUTURE WORK

Six models that allow predicting the relevance of an article regarding adjuvants of antibiotics have been developed throughout this work. All models have been tested using *CV* with 5-folds. The scores of the selected models, obtained through the sklearn package, were low, perhaps due to the redundancy of the features. Therefore, feature selection and optimization have been applied. The comparison of the results of six different models shows that the best results obtained for each metric were: Accuracy = 0.8461, Recall = 0.6170, F1 score = 0.69904 in the LR - 1 model, and Precision = 0.793 in the RF - 2. Analyzing all metrics carefully, it is possible to conclude that the best model for classifying relevant articles would be the *LR* –1. It offers an accuracy of 0.8461, a recall of 0.6170, an f1-score of 0.6904, and a precision of 0.7837. These results show that this model is the best at correctly predicting relevant articles, as proven by the higher recall score compared to other models. The model also wrongly predicted irrelevant articles, which led to a lower f1-score. The high accuracy shows that this model is capable of predicting correctly relevant and irrelevant articles. Regarding precision, this model is the second best, having more *FP* than the *RF* –2 model. Even though this model is not perfect, the model would classify more articles correctly than the other models. Also, this model will have fewer results, which is preferable because the model does not misclassify the true relevant articles in the dataset, even if that costs some of the *FP*.

The backend and frontend were developed using Python Django and *DRF* to construct the *API* and React for the user interface. ***BIOFILMad*** - *Catalog of antimicrobial adjuvants to tackle biofilms* is an application with filtered articles that can help to find new compounds to

develop anti-biofilm therapies. Researchers spend hours searching in literature compounds that have already been tested. This application can help with this liability. It has a unique feature that shows the relations between the antibiofilm compounds, antimicrobials, and microorganisms/diseases, helping researchers saving time. Instead of spending hours searching in literature what has already been tested, this feature can show that information to the researchers in a naive way. It is essential to consider that the *ML* model was trained to consider everything that was tested with antibiotics or not. Hence, in addition to antibiotics, there are articles about other antimicrobials that have already been tested, which can now be analyzed and help researchers find new combinations with antibiotics.

Regarding limitations, collecting relevant data is challenging as the literature is scattered, and there is not much relevant information on this topic. Also, the fact that the information provided to the model to learn how to assign a label is diverse can make the model itself not so accurate. This can be an analogy to the problem faced nowadays. It is difficult to find articles on antibiotics adjuvants; thus, the model also has some difficulties classifying them as relevant.

Also, devising a way to show the content in a clean and straightforward manner that facilitates the study of anti-biofilm therapies was challenging. The key was using the ScispaCy library with the *NER* by classifying the eligible entities for biomedical research. However, we faced issues with the task of identifying entities, as some abbreviations were present in the text, and ScispaCy identified them as entities. Hence, as a solution, we converted most to their extended form, but not all were possible to convert. This led to abbreviations being shown on the user interface instead of the extended and more straightforward form. Likewise, the Scispacy classified the same entity as different if the letters were uppercase or lowercase. The solution was converting all entities to lowercase using Inflect, and with that, much of the duplicated entities disappeared, making cleaner content.

The main goal proposed for this thesis, the development of an application that helps researchers find anti-biofilm therapy studies, was accomplished. However, some steps can be improved in the future:

- Include more articles in the training and test sets to help the models learn how to classify articles correctly and efficiently;
- Implement more features with a different perspective or even create completely different features from the beginning and see their impact on the model's performance;
- Add more articles to the database to give the user more information;
- Improve the aggregation feature, maybe by using AI to have a perspective through the abstract if the result of the article's research was positive or negative;

Altogether this application can help researchers save time researching anti-biofilm therapies. This potentiates the discovery of compounds that potentiate existing antimicrobials, such as antibiotics, in less time than nowadays, saving countless lives.

BIBLIOGRAPHY

[1] J. Gauthier, A. T. Vincent, S. J. Charette, and N. Derome, "A brief history of bioinformatics," *Briefings in Bioinformatics*, vol. 20, no. 6, pp. 1981–1996, 2019.

[2] K. G. Srinivasan, G. M. Siddesh, and S. R. Manisekhar, *Statistical Modelling and Machine Learning Principles for Bioinformatics Techniques, Tools, and Applications*. 2020.

[3] N. M. Luscombe, D. Greenbaum, and M. Gerstein, "What is bioinformatics? A proposed definition and overview of the field," *Methods of Information in Medicine*, vol. 40, no. 4, pp. 346–358, 2001.

[4] J. Xiong, *Essential Bioinformatics*. 2006.

[5] M. Whithorn and B. Marklyn, *Inside Relational Databases with Examples in ACCESS*. Springer Science+Business Media, 2007.

[6] I. Matinolli, "Designing , Implementing and Evaluating a Database for a Software Testing Team," pp. 1–82, 2016.

[7] S. Yu, "ACID Properties in Distributed Databases," *Advanced eBusiness Transactions for B2B-Collaborations*, p. 17, 2009.

[8] R. Tiffany, *SQL Server CE Database - Development with the .NET Compact Framework*. Apress, 2003.

[9] R. Apweiler, A. Bairoch, and C. H. Wu, "Protein sequence databases," *Current Opinion in Chemical Biology*, vol. 8, no. 1, pp. 76–80, 2004.

[10] C. Kanz, P. Aldebert, N. Althorpe, W. Baker, A. Baldwin, K. Bates, P. Browne, A. van den Broek, M. Castro, G. Cochrane, K. Duggan, R. Eberhardt, N. Faruque, J. Gamble, F. Garcia Diez, N. Harte, T. Kulikova, Q. Lin, V. Lombard, R. Lopez, R. Mancuso, M. McHale, F. Nardone, V. Silventoinen, S. Sobhany, P. Stoehr, M. A. Tuli,

K. Tzouvara, R. Vaughan, D. Wu, W. Zhu, and R. Apweiler, "The EMBL nucleotide sequence database," *Nucleic Acids Research*, vol. 33, no. DATABASE ISS., pp. 29–33, 2005.

[11] R. A. Laskowski, "Protein structure databases," *Molecular Biotechnology*, vol. 48, no. 2, pp. 183–198, 2011.

[12] D. Schomburg and I. Schomburg, "Enzyme databases.," *Methods in molecular biology (Clifton, N.J.)*, vol. 609, pp. 113–128, 2010.

[13] J. Hubble, J. Demeter, H. Jin, M. Mao, M. Nitzberg, T. B. Reddy, F. Wymore, Z. K. Zachariah, G. Sherlock, and C. A. Ball, "Implementation of GenePattern within the Stanford Microarray Database," *Nucleic Acids Research*, vol. 37, no. SUPPL. 1, pp. 898–901, 2009.

[14] D. Peter, "Pathway databases: Making chemical and biological sense of the genomic data flood," *Chemistry and Biology*, vol. 20, no. 5, pp. 629–635, 2013.

[15] M. A. Miller, "Chemical database techniques in drug discovery," *Nature Reviews Drug Discovery*, vol. 1, no. 3, pp. 220–227, 2002.

[16] A. Y. Gasparyan, L. Ayvazyan, and G. D. Kitas, "Multidisciplinary bibliographic databases," *Journal of Korean Medical Science*, vol. 28, no. 9, pp. 1270–1275, 2013.

[17] Vijay Kotu and B. Deshpande, "Text mining. Data Science," pp. 281–305, 2019.

[18] V. Gupta and G. S. Lehal, "A Survey of Text Mining Techniques and Applications - Volume 1, No. 1, August 2009 - JETWI," *Journal of Emerging Technologies in Web Intelligence*, vol. 1, no. 1, pp. 60–76, 2009.

[19] P. O. Larsen and M. von Ins, "The rate of growth in scientific publication and the decline in coverage provided by science citation index," *Scientometrics*, vol. 84, no. 3, pp. 575–603, 2010.

[20] G. Miner, J. Elder, IV, A. Fast, T. Hill, R. Nisbet, and D. Delen, "Practical Text Mining and Statistical Analysis for Non-Structured Text Data Applications. The Seven Practice Areas of Text Analytics," *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications*, pp. 29–41, 2012.

[21] D. Westergaard, H. H. Stærfeldt, C. Tønsberg, L. J. Jensen, and S. Brunak, "A comprehensive and quantitative comparison of text-mining in 15 million full-text articles versus their corresponding abstracts," *PLoS Computational Biology*, vol. 14, no. 2, pp. 1–16, 2018.

[22] W. R. Hersh, *Information retrieval: a health and biomedical perspective*. 2008.

[23] K. Sharma, G. Shrivastava, and V. Kumar, "Web mining: Today and tomorrow," *ICECT 2011 - 2011 3rd International Conference on Electronics Computer Technology*, vol. 1, pp. 399–403, 2011.

[24] S. Sarawagi, *Information Extraction*. Now Publishers Inc, 2008.

[25] J. R. Hobbs and E. Riloff, "Chapter 21: Information Extraction," *Handbook of Natural Language Processing*, pp. 511–532, 2010.

[26] P. Zweigenbaum, D. Demner-fushman, H. Yu, and K. B. Cohen, "Frontiers of biomedical text mining: Current progress," *Briefings in Bioinformatics*, vol. 8, no. 5, pp. 358–375, 2007.

[27] R. Rodriguez-Esteban, "Biomedical text mining and its applications," *PLoS Computational Biology*, vol. 5, no. 12, pp. 1–5, 2009.

[28] M. Krallinger and A. Valencia, "Text-mining and information-retrieval services for molecular biology," *Genome Biology*, vol. 6, no. 7, 2005.

[29] A. Lourenço, R. Carreira, S. Carneiro, P. Maia, D. Glez-Peña, F. Fdez-Riverola, E. C. Ferreira, I. Rocha, and M. Rocha, "@Note: A workbench for Biomedical Text Mining," *Journal of Biomedical Informatics*, vol. 42, no. 4, pp. 710–720, 2009.

[30] A. Lourenço, R. Carreira, D. Glez-Peña, J. R. Méndez, S. Carneiro, L. M. Rocha, F. Díaz, E. C. Ferreira, I. Rocha, F. Fdez-Riverola, and M. Rocha, "BioDR: Semantic indexing networks for biomedical document retrieval," *Expert Systems with Applications*, vol. 37, no. 4, pp. 3444–3453, 2010.

[31] D. Sarkar, *Text Analytics with Python (Chaper 5)*. 2019.

[32] G. Miner, J. Elder, IV, A. Fast, T. Hill, R. Nisbet, and D. Delen, "Text Classification and Categorization," *Practical Text Mining and Statistical Analysis for Non-structured Text Data Applications*, pp. 881–892, 2012.

[33] B. Bengfort, R. Bilbro, and T. Ojeda, *Applied text analysis with Python: enabling language-Aware data products with machine learning*.

[34] K. Kowsari, K. J. Meimandi, M. Heidarysafa, S. Mendu, L. Barnes, and D. Brown, "Text classification algorithms: A survey," *Information (Switzerland)*, vol. 10, no. 4, pp. 1–68, 2019.

[35] P. H. Seo, Z. Lin, S. Cohen, X. Shen, and B. Han, "Hierarchical Attention Networks," *ArXiv*, pp. 1480–1489, 2016.

[36] G. Ignatow and R. Mihalcea, *Text mining: A guidebook for the social sciences*. 2016.

[37] D. Virmani and S. Taneja, *A text preprocessing approach for efficacious information retrieval*, vol. 669. Springer Singapore, 2019.

[38] L. Hickman, S. Thapa, L. Tay, M. Cao, and P. Srinivasan, "Text Preprocessing for Text Mining in Organizational Research: Review and Recommendations," *Organizational Research Methods*, vol. 25, no. 1, pp. 114–146, 2022.

[39] N. Hardeniya, J. Perkins, D. Chopra, N. Joshi, and I. Mathur, *Natural language processing: python and NLTK*. Packt, 2016.

[40] K. Tomanek, J. Wermter, and U. Hahn, "Sentence and Token Splitting Based On Conditional Random Fields," *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, p. 57, 2007.

[41] M. Kannan, S., Gurusamy, V., Vijayarani, S., Ilamathi, J. Nithya, "Preprocessing Techniques for Text Mining - An Overview," *International Journal of Computer Science Communication Networks*, vol. 5, no. October 2014, pp. 7–16, 2015.

[42] M. Kannan, S., Gurusamy, V., Vijayarani, S., Ilamathi, J. Nithya, "Preprocessing Techniques for Text Mining," *International Journal of Computer Science Communication Networks*, vol. 5, no. October 2014, pp. 7–16, 2015.

[43] V. S and J. R, "Text Mining: open Source Tokenization Tools – An Analysis," *Advanced Computational Intelligence: An International Journal (ACII)*, vol. 3, no. 1, pp. 37–47, 2016.

[44] J. O. Contreras, S. Hilles, and Z. B. Abubakar, "Automated Essay Scoring with Ontology based on Text Mining and NLTK tools," *2018 International Conference on Smart Computing and Electronic Enterprise, ICSCEE 2018*, pp. 1–6, 2018.

[45] S. Bird and E. Loper, "NLTK: The natural language toolkit," *Association for Computational Linguistics.*, 2004.

[46] F. Wolf, T. Poggio, and P. Sinha, "Bags of Words," *Bags of Words*, pp. 70–87, 2006.

[47] E. L. B. Stevens, E. Klein, *Natural Language Processing with Python*. 1375.

[48] G. Gosal and N. Sharma, "A Review of Approaches for Entity Recognition from Biological Text," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 14, no. 12, 2016.

[49] A. Mansouri, L. S. Affendey, and A. Mamat, "Named Entity Recognition Approaches," *Journal of Computer Science*, vol. 8, no. 2, pp. 339–344, 2008.

[50] G. Shobha and S. Rangaswamy, *Machine Learning*, vol. 38. Elsevier B.V., 1 ed., 2018.

[51] X. Y. Liu and Yong, *Machine Learning.* 2013.

[52] M. Batta, "Machine Learning Algorithms - A Review ," *International Journal of Science and Research (IJ*, vol. 9, no. 1, pp. 381–undefined, 2020.

[53] P. Harrington, *Machine learning in action*. Manning Publications, 2012.

[54] E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and O. E. Ajibuwa, "Machine learning for email spam filtering: review, approaches and open research problems," *Heliyon*, vol. 5, no. 6, 2019.

[55] O. Bousquet, U. von Luxburg, and Gunnar Rätsch, *Advanced Lectures on Machine Learning*, vol. 30. 1959.

[56] T. Hastie, R. Tibshirani, and J. Friedman, "Unsupervised learning.," *The Elements of Statistical Learning*, pp. 485–585, 2009.

[57] G. L. T. C and M. Shang, *Hands on Supervised Learning With Python*.

[58] T. Jo, *Machine Learning Foundations*. 2021.

[59] L. P. Kaelbling, M. L. Littman, and A. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, pp. 237–285, 1996.

[60] R. S. Sutton and A. G. Barto, *Reinforcement Learning, second edition: An Introduction*.

[61] M. Goadrich, L. Oliphant, and J. Shavlik, "Gleaner: Creating ensembles of first-order clauses to improve recall-precision curves," *Machine Learning*, vol. 64, no. 1-3, pp. 231–261, 2006.

[62] H. Blockeel, K. Kersting, S. Nijssen, and F. Zelezny, *Machine Learning and Knowledge Discovery in Databases*. No. September, 2012.

[63] J. Fan, S. Upadhye, and A. Worster, "Understanding receiver operating characteristic (ROC) curves," *Canadian Journal of Emergency Medicine*, pp. 19–20, 2006.

[64] A. P. Yonelinas and C. M. Parks, "Receiver Operating Characteristics (ROCs) in Recognition Memory: A Review," *Psychological Bulletin*, vol. 133, no. 5, pp. 800–832, 2007.

[65] E. Gabrilovich, "Feature generation for textual information retrieval using world knowledge," *ACM SIGIR Forum*, vol. 41, no. 2, pp. 123–123, 2007.

[66] S. Doraisamy, S. Golzari, N. M. Norowi, M. N. B. Sulaiman, and N. I. Udzir, "A study on feature selection and classification techniques for automatic genre classification of traditional malay music," *ISMIR 2008 - 9th International Conference on Music Information Retrieval*, pp. 331–336, 2008.

[67] "Django, the web framework for perfectionists with deadlines." https://www.djangoproject.com/. Accessed: 2022/01/21.

[68] "Express, fast, unopinionated, minimalist web framework for node.js." https://docs.djangoproject.com/en/4.1/topics/db/models/. Accessed: 2022/01/21.

[69] "Flask web development, one drop at a time." https://flask.palletsprojects.com/en/1.1.x/. Accessed: 2022/01/21.

[70] "React.js, a javascript library for building user interfaces." https://reactjs.org/. Accessed: 2022/01/21.

[71] "Angular.js , the modern web developer's platform." https://angular.io/. Accessed: 2022/01/21.

[72] "jquery.js, write less, do more." https://jquery.com/. Accessed: 2022/01/21.

[73] "Django - models." https://docs.djangoproject.com/en/4.1/topics/db/models/. Accessed: 2022/09/22.

[74] "Django - views." https://docs.djangoproject.com/en/4.1/topics/http/views/. Accessed: 2022/09/22.

[75] M. Alchin, "Pro django: Second edition," *Pro Django: Second Edition*, vol. 9781430258100, pp. 1–273, 2013.

[76] "Django - urls." https://docs.djangoproject.com/en/4.1/topics/http/urls/. Accessed: 2022/09/22.

[77] D. Rubio, *Beginning Django*. 2017.

[78] "Django rest framework." https://www.django-rest-framework.org/. Accessed: 2022/09/22.

[79] A. Satyal and P. W. Application, "Designing and developing a website with reactjs,"

[80] "React - components and props." https://reactjs.org/docs/components-and-props.html. Accessed: 2022/09/22.

[81] "React - introducing hooks." https://reactjs.org/docs/hooks-intro.html. Accessed: 2022/09/22.

[82] G. Kapoor, S. Saigal, and A. Elongavan, "Action and resistance mechanisms of antibiotics: A guide for clinicians," *Journal of Anaesthesiology Clinical Pharmacology*, pp. 300–305, 2017.

[83] C. L. Ventola, "The Antibiotics Resistance Crisis Part 1: Causes and Threats," *Comprehensive Biochemistry*, pp. 277–283, 2015.

[84] H. Douafer, V. Andrieu, O. Phanstiel, and J. M. Brunel, "Antibiotic Adjuvants: Make Antibiotics Great Again!," *Journal of Medicinal Chemistry*, vol. 62, no. 19, pp. 8665–8681, 2019.

[85] D. S. Pontes, R. S. A. de Araujo, N. Dantas, L. Scotti, M. T. Scotti, R. O. de Moura, and F. J. B. Mendonca-Junior, "Genetic Mechanisms of Antibiotic Resistance and the Role of Antibiotic Adjuvants," *Current Topics in Medicinal Chemistry*, vol. 18, no. 1, pp. 42–74, 2018.

[86] I. Ghai and S. Ghai, "Understanding antibiotic resistance via outer membrane permeability," *Infection and Drug Resistance*, vol. 11, pp. 523–530, 2018.

[87] S. Džidić, J. Šušković, and B. Kos, "Antibiotic resistance mechanisms in bacteria: Biochemical and genetic aspects," *Food Technology and Biotechnology*, vol. 46, no. 1, pp. 11–21, 2008.

[88] R. Mirzaei, R. Mohammadzadeh, M. Y. Alikhani, M. Shokri Moghadam, S. Karampoor, S. Kazemi, A. Barfipoursalar, and R. Yousefimashouf, "The biofilm-associated bacterial infections unrelated to indwelling devices," *IUBMB Life*, vol. 72, no. 7, pp. 1271–1285, 2020.

[89] M. Jamal, W. Ahmad, S. Andleeb, F. Jalil, M. Imran, M. A. Nawaz, T. Hussain, M. Ali, M. Rafiq, and M. A. Kamil, "Bacterial biofilm and associated infections," *Journal of the Chinese Medical Association*, vol. 81, no. 1, pp. 7–11, 2018.

[90] T. Sen, S. Karmakar, and R. Sarkar, "Chapter 15 - Evaluation of Natural Products against Biofilm-Mediated Bacterial Resistance," *Evidence-Based Validation of Herbal Medicine*, pp. 321–338, 2015.

[91] A. Kumar, A. Alam, M. Rani, N. Z. Ehtesham, and S. E. Hasnain, "Biofilms: Survival and defense strategy for pathogens," *International Journal of Medical Microbiology*, vol. 307, no. 8, pp. 481–489, 2017.

[92] M. S. Aparna and S. Yadav, "Biofilms: Microbes and disease," *Brazilian Journal of Infectious Diseases*, vol. 12, no. 6, pp. 526–530, 2008.

[93] P. S.Stewart, "Mechanisms of antibiotic resistance in bacterial biofilms," *International Journal of Medical Microbiology*, pp. 107–113, 2002.

[94] T.-f. Mah, "Biofilm-specific antibiotic resistance," *Future microbiology*, pp. 1061–1072, 2012.

[95] R. M. Donlan and J. W. Costerton, "Biofilms: Survival mechanisms of clinically relevant microorganisms," *Clinical Microbiology Reviews*, vol. 15, no. 2, pp. 167–193, 2002.

[96] P. S. Stewart and J. W. Costerton, "Antibiotic resistance of bacteria in biofilms," *Lancet*, vol. 358, no. 9276, pp. 135–138, 2001.

[97] S. Dincer, F. M. Uslu, and A. Delik, "Antibiotic Resistance in Biofilm," *Bacterial Biofilms.*, 2020.

[98] Z. A. Mirani, S. Urooj, A. Ullah, M. N. Khan, N. Rauf, A. Mehmood, W. Fenghuan, I. A. Shaikh, and A. B. Khan, "Phenotypic Heterogeneity in Biofilm Consortia of E. coli," *Microbiology (Russian Federation)*, vol. 90, no. 2, pp. 237–246, 2021.

[99] W. A. Weigel and P. Dersch, "Phenotypic heterogeneity: a bacterial virulence strategy," *Microbes and Infection*, vol. 20, no. 9-10, pp. 570–577, 2018.

[100] M. A. Kohanski, D. J. Dwyer, and J. J. Collins, "How antibiotics kill bacteria: From targets to networks," *Nature Reviews Microbiology*, vol. 8, no. 6, pp. 423–435, 2010.

[101] E. Silva, R. Monteiro, T. Grainha, D. Alves, M. O. Pereira, and A. M. Sousa, "Fostering Innovation in the Treatment of Chronic Polymicrobial Cystic Fibrosis-Associated Infections Exploring Aspartic Acid and Succinic Acid as Ciprofloxacin Adjuvants," *Frontiers in Cellular and Infection Microbiology*, vol. 10, no. August, pp. 1–14, 2020.

[102] G. D. Wright, "Antibiotic Adjuvants: Rescuing Antibiotics from Resistance," *Trends in Microbiology*, vol. 24, no. 11, pp. 862–871, 2016.

[103] B. Nassima, B. Nassima, and K. Riadh, "Antimicrobial and antibiofilm activities of phenolic compounds extracted from Populus nigra and populus alba buds (Algeria)," *Brazilian Journal of Pharmaceutical Sciences*, vol. 55, 2019.

[104] L. Lu, W. Hu, Z. Tian, D. Yuan, G. Yi, Y. Zhou, Q. Cheng, J. Zhu, and M. Li, "Developing natural products as potential anti-biofilm agents," *Chinese Medicine (United Kingdom)*, vol. 14, no. 1, pp. 1–17, 2019.

[105] A. Rosato, S. Sblano, L. Salvagno, A. Carocci, M. L. Clodoveo, F. Corbo, and G. Fracchiolla, "Anti-biofilm inhibitory synergistic effects of combinations of essential oils and antibiotics," *Antibiotics*, vol. 9, no. 10, pp. 1–13, 2020.

[106] J. H. Hwang, H. Choi, I. S. Hwang, A. R. Kim, E. R. Woo, and D. G. Lee, "Synergistic antibacterial and antibiofilm effect between (+)-medioresinol and antibiotics in vitro," *Applied Biochemistry and Biotechnology*, vol. 170, no. 8, pp. 1934–1941, 2013.

[107] "Biomedical text mining tool for document triage." https://gitlab.bio.di.uminho.pt/biotextminingpy/biotmpy. Accessed: 2022/09/22.

[108] P. J. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, *et al.*, "Biopython: freely available python tools for computational molecular biology and bioinformatics," *Bioinformatics*, vol. 25, no. 11, pp. 1422–1423, 2009.

[109] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[110] G. Van Rossum, *The Python Library Reference, release 3.8.2*. Python Software Foundation, 2020.

[111] P. Biofilms, M. M. Maiden, M. A. Hunt, M. P. Zachos, J. A. Gibson, M. E. Hurwitz, M. H. Mulks, and M. Waters, "crossm Triclosan Is an Aminoglycoside Adjuvant for Eradication of," pp. 1–16, 2018.

[112] "Spacy - industrial-strength natural language processing." https://spacy.io/. Accessed: 2022/09/25.

[113] No Starch Press, 2020.

[114] B. Srinivasa-Desikan, *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd, 2018.

[115] "Spacy models for biomedical text processing." https://allenai.github.io/scispacy/. Accessed: 2022/09/25.

[116] "Axios promise based http client for the browser and node.js." https://axios-http.com/. Accessed: 2022/09/25.

[117] A. Lauret, *The Design of Web APIs*. 1 ed., 2019.

[118] "Redhat - what is an api?." https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces. Accessed: 2022/10/12.

[119] R. T. Fielding and R. N. Taylor, "Principled design of the modern web architecture," *Proceedings - International Conference on Software Engineering*, pp. 407–416, 2000.

[120] "codecademy - what is rest?." https://www.codecademy.com/article/what-is-rest. Accessed: 2022/10/12.

[121] "Redhat - what is a rest api?." https://www.redhat.com/en/topics/api/what-is-a-rest-api. Accessed: 2022/10/12.

[122] F. Doglio, *REST API Development with Node.js*. 2nd ed. ed., 2018.

[123] M. Neumann, D. King, I. Beltagy, and W. Ammar, "ScispaCy: Fast and robust models for biomedical natural language processing," *BioNLP 2019 - SIGBioMed Workshop on Biomedical Natural Language Processing, Proceedings of the 18th BioNLP Workshop and Shared Task*, vol. 0, pp. 319–327, 2019.

Appendices

A

SUPPLEMENTARY MATERIAL

A.1 APIS

APIs enable applications to exchange data and functionality easily and securely, simplifying software development. More precisely, an *API* is a set of rules explaining how the applications and computers communicate with each other, designed to be used by a computer and applications. The *API* connects the web server and the application, becoming a layer that processes the data transfer between the systems [117].

The *API* works essentially in four steps. The first step comprises the application making a request, thus initiating an *API* call to retrieve the desired data in the request. The *API* will receive this request and will make a call to the web server. This web server will retrieve the request information and will send a response to the *API*. The *API* now has the user-requested information, and this data will be transferred to the application. This data transfer will depend on the web service being used, but the process of request and response happens through an *API* [117].

These data transfers have to follow a specific protocol to facilitate information exchange. Different protocols have specific rules with the available commands and data types. Examples of *API* protocols are listed below:

- ***SOAP*** is a protocol built with *XML* , that sends and receives data through *SMTP* and *HTTP* [118].

- **XML-RPC** is a protocol that uses a specific format of *XML* to transfer data.
- is similar to *XML-RPC*, as it uses remote procedure calls, though using the *JSON* format to transfer data.
- **REST** was intrduced by Roy Fielding [119] in his dissertation thesis in 2000. REST (or RESTful *API*) is characterized by not following a protocol or a standard, making it easier for systems to communicate with each other [118] [120]. Instead, they have the following architectural constraints:
  - **Client-Server Architecture –** This means that the implementation of the client and the server are independent of each other. The client side does not have to know how the server-side is implemented and vice versa. The only thing each side knows is the format of the messages sent between the two [121] [120] [122].

    The request made by the clients are intended to retrieve or modify resources, and the servers will send responses to those requests. This request generally consists of [120]:

    * An *HTTP* verb defining the operation to perform (GET, POST, PUT or DELETE).
    * A header to pass information with the *HTTP* request.
    * A path to the specific resource.
    * The body of the message encompassing data (optional).
  - **Statelessness –** Client-server communication is stateless; this means that the client does not know in which state the server is and vice versa, each request is separate and unconnected, so there is no information stored about the client [121] [120] [122].
  - **Cacheable –** The objective of caching the data is to save the server to generate the same response more than once. This reduces the client-server interactions by reducing server load and increasing the velocity of the response [121] [122].

- **Layered System –** This system has organized servers, each responsible for a single purpose (i.e., security, storing data, etc.) involved in retrieving requested information. These layers are invisible to the client, and the client does not know to which server they are connected to [121] [122].
- **Uniform Interface –** this interface provides a standard form of communication between client and server, decoupling the interface from the implementation. There are four principles of this interface [122] [121]:
    - **Resource-based:** the requested resources are identifiable (using URIs) and are separated from the representations returned to the client. For example, the server will not send the database, instead, it will send a representation of database records in a format like *JSON* [119].
    - **Manipulation of resources through these representations:** clients can manipulate the resource on the server through the representation they receive (if they have permission to) [121].
    - **Self-descriptive messages:** this message returned to the client has sufficient information to describe how to process it [121].
    - ***HATEOAS*:** means that hypertext/hypermedia is available. After access to a resource, the client can go to the other links available and get to other resources [121].
- **Code on demand (optional) -** enhancing client functionality by sending executable code, when requested, from the server to the client [122] [121].

A.2 ENTITY RELATIONSHIP DIAGRAM

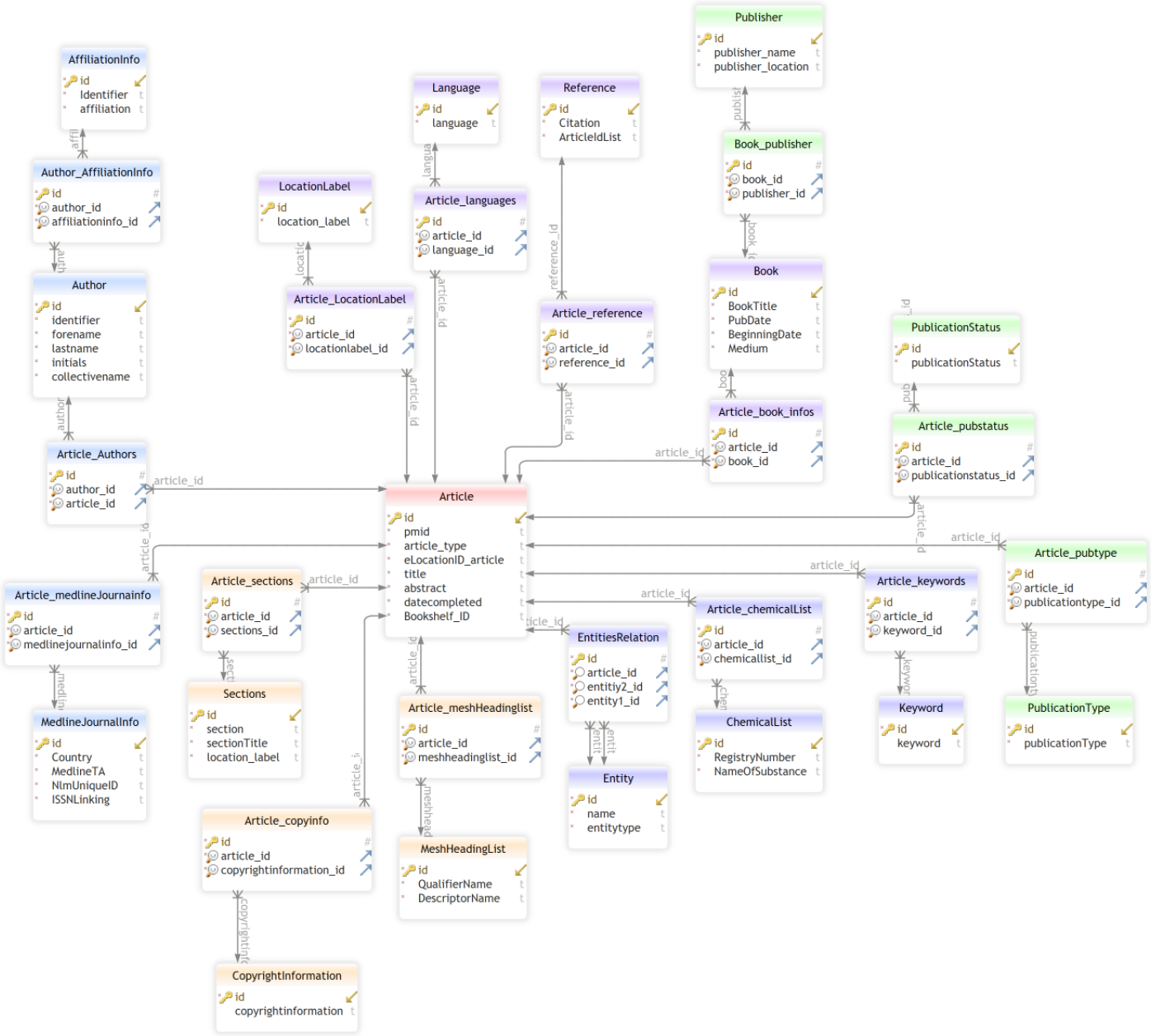


Figure 28: Entity Relationship Diagram of database tables.

A.3 DJANGO REST FRAMEWORK VIEWSETS

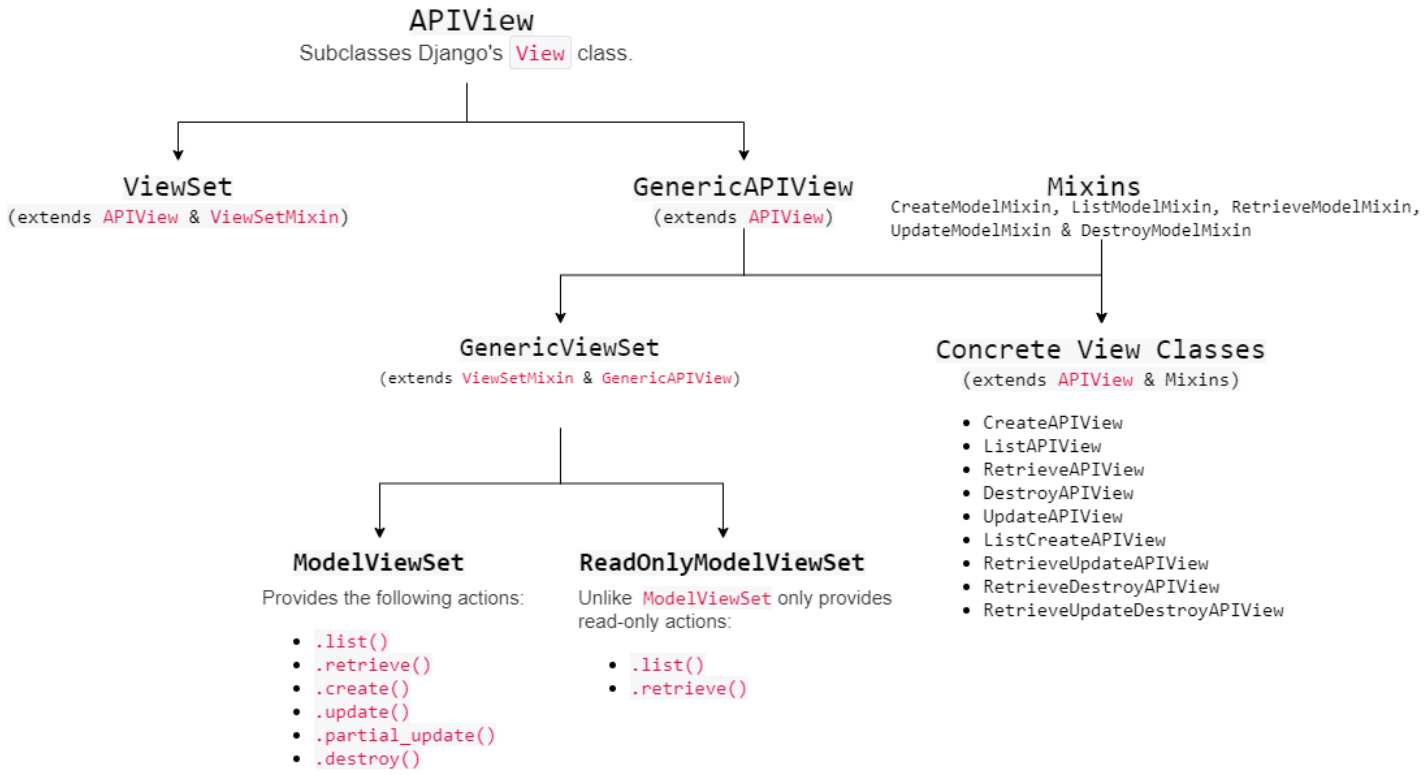


Figure 29: Django REST Framework view inheritance diagram.

A.4 SCISPACY TWO MAIN PACKAGES

Table 10: Description and vocabulary statistics for the two main packages.

Mark Neumann, Daniel King, Iz Beltagy, Waleed Ammar [123]

Model	Vocab Size	Vector Count	Min Word Freq	Min Doc Freq
en_core_sci_sm	58,338	0	50	5
en_core_sci_md	101,678	98,131	20	5

A.5 SCISPACY OTHER PACKAGES

Table 11: Models of ScispaCy.

Mark Neumann, Daniel King, Iz Beltagy, Waleed Ammar [123]

Model	Description	Entity Types
en_core_sci_scibert	A full spaCy pipeline for biomedical data with a ~785k vocabulary and allenai/scibert-base as the transformer model.	-
en_core_sci_lg	A full spaCy pipeline for biomedical data with a larger vocabulary and 600k word vectors.	-
en_ner_craft_md	A spaCy NER model trained on the CRAFT corpus.	GGP, SO, TAXON, CHEBI, GO, CL
en_ner_jnlpba_md	A spaCy NER model trained on the JNLPBA corpus.	DNA, CELL_TYPE, CELL_LINE, RNA, PROTEIN
en_ner_bc5cdr_md	A spaCy NER model trained on the BC5CDR corpus.	DISEASE, CHEMICAL
en_ner_bionlp13cg_md	A spaCy NER model trained on the BIONLP13CG corpus.	AMINO_ACID, CANCER, CELL, ORGANISM, TISSUE, MULTI-TISSUE_STRUCTURE, ORGAN, SIMPLE_CHEMICAL, GENE_OR_GENE_PRODUCT, DEVELOPING_ANATOMICAL_STRUCTURE, ANATOMICAL_SYSTEM, CELLULAR_COMPONENT, IMMATERIAL_ANATOMICAL_ENTITY, ORGANISM_SUBDIVISION, ORGANISM_SUBSTANCE, PATHOLOGICAL_FORMATION